設計與實作路燈管理系統：配置與佈署

Design and Implementation of a Street Light Management System: Configuration and Deployment

REPORTER: GUAN-SHENG CHEN

ADVISOR: PROF. CHU-SING YANG

DATE: 2019/07/15

Outline

- Introduction
- Background and Related Works
- System Design
- Experiment and Verification
- Conclusion and Future Works

Outline

- Introduction
- Background and Related Works
- System Design
- Experiment and Verification
- Conclusion and Future Works

Introduction

- Through integrating various **sensors** and **embedded devices** provides the opportunity to transform **streetlights** into a multifunctional digital urban platform.
- However, existing systems are far less **flexible** and developed without **security and privacy** protections.
- Our work implements a streetlight management system which is
  - **Highly scalable**
    - modular, rapid deployment, high resource utilization
  - **Security oriented**
    - transmit data with authentication (permission) and encryption
  - **User friendly**
    - easy-to-deploy, real-time feedback, multiuser accessibility

Outline

- Introduction
- Background and Related works
- System Design
- Experiment and Verification
- Conclusion and Future works

Related Works

- Re-Imagining Streetlight Infrastructure as a Digital Urban Platform ... Journal of Urban Technology, 2017
- Smart IP – **central management system** for public lighting in Portugal ... CIRED Journal, 2017
- Design of Smart LED Streetlight System for Smart City With **Web-Based Management System** … IEEE Sensors Journal, 2017
- Smart Outdoor Light Desktop **Central Management System** ... IEEE Intelligent Transportation Systems Magazine, 2018
- SSL: Smart Street Lamp Based on Fog Computing for Smarter Cities ... IEEE Transactions on Industrial Informatics, 2018
- Our proposed system pays more attention to
  - Manageability and Scalability of **deployment**
  - Encryption and Authentication for **protecting communications**

Container vs. Virtual Machine

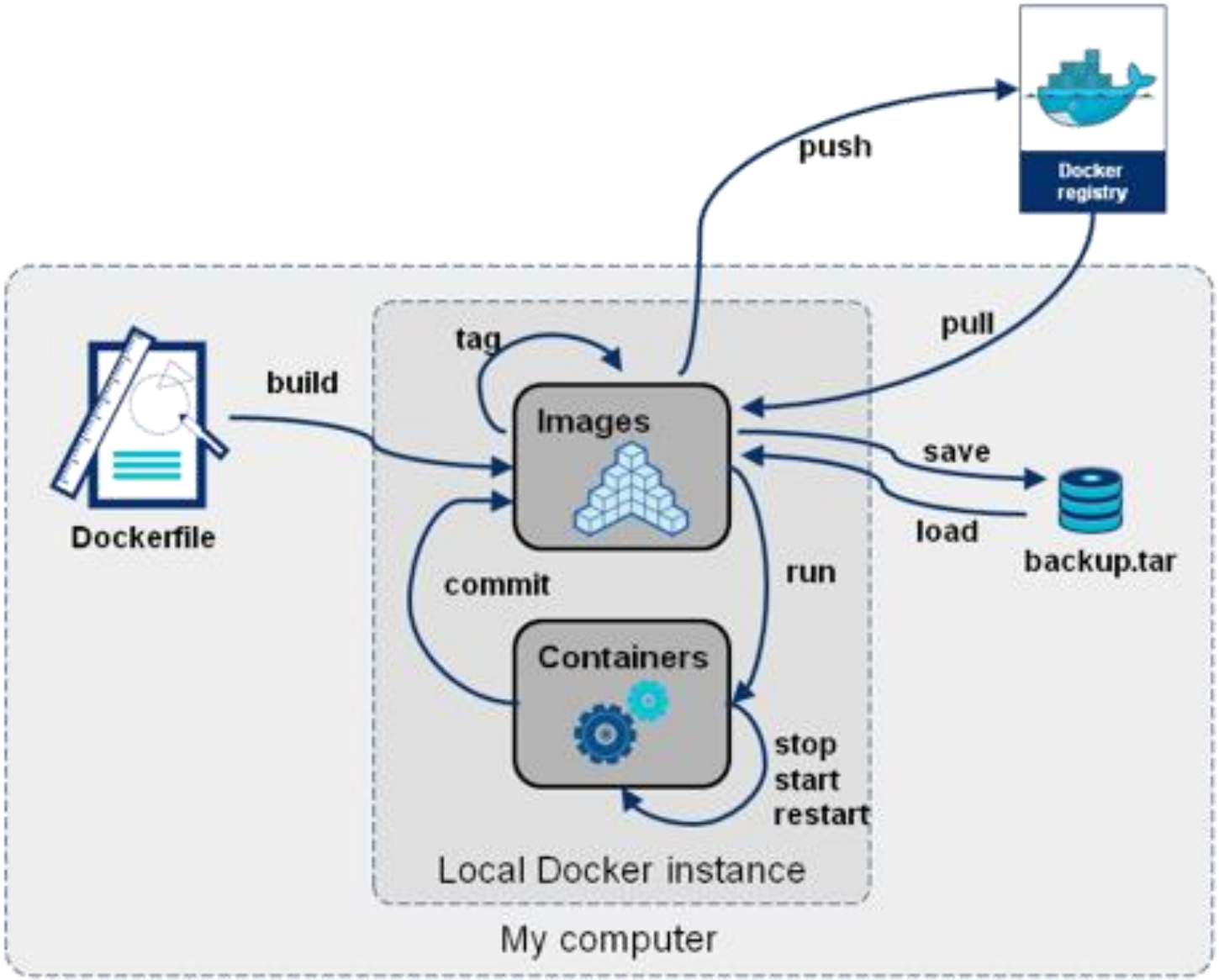
- **Docker** is a high-level containerization API

Table 2.1: Comparison of Docker and Virtual Machines [2]

Feature	Docker	Virtual Machines
Start time	<50 ms	30 - 45 seconds
Stop time	<50 ms	5 - 10 seconds
System Overhead	No overhead	Overhead due to hypervisor
Storage space	Tens of MBs in size	Tens of GBs in size
Scalability	Highly scalable	Not easily scalable
CPU load when idle	Normal	~1.5% more than docker
Isolation	Less isolation due to software virtualization	More isolation due to hardware virtualization
Network round trip latency	~75 µs	~60 µs
I/O throughput (Read and Write)	100000 I/Os per second	50000 I/Os per second

"Netra: Enhancing iot security using nfvbased edge traffic analysis," IEEE Sensors Journal, 2019.

Docker Architecture



Containerization in IoT environment

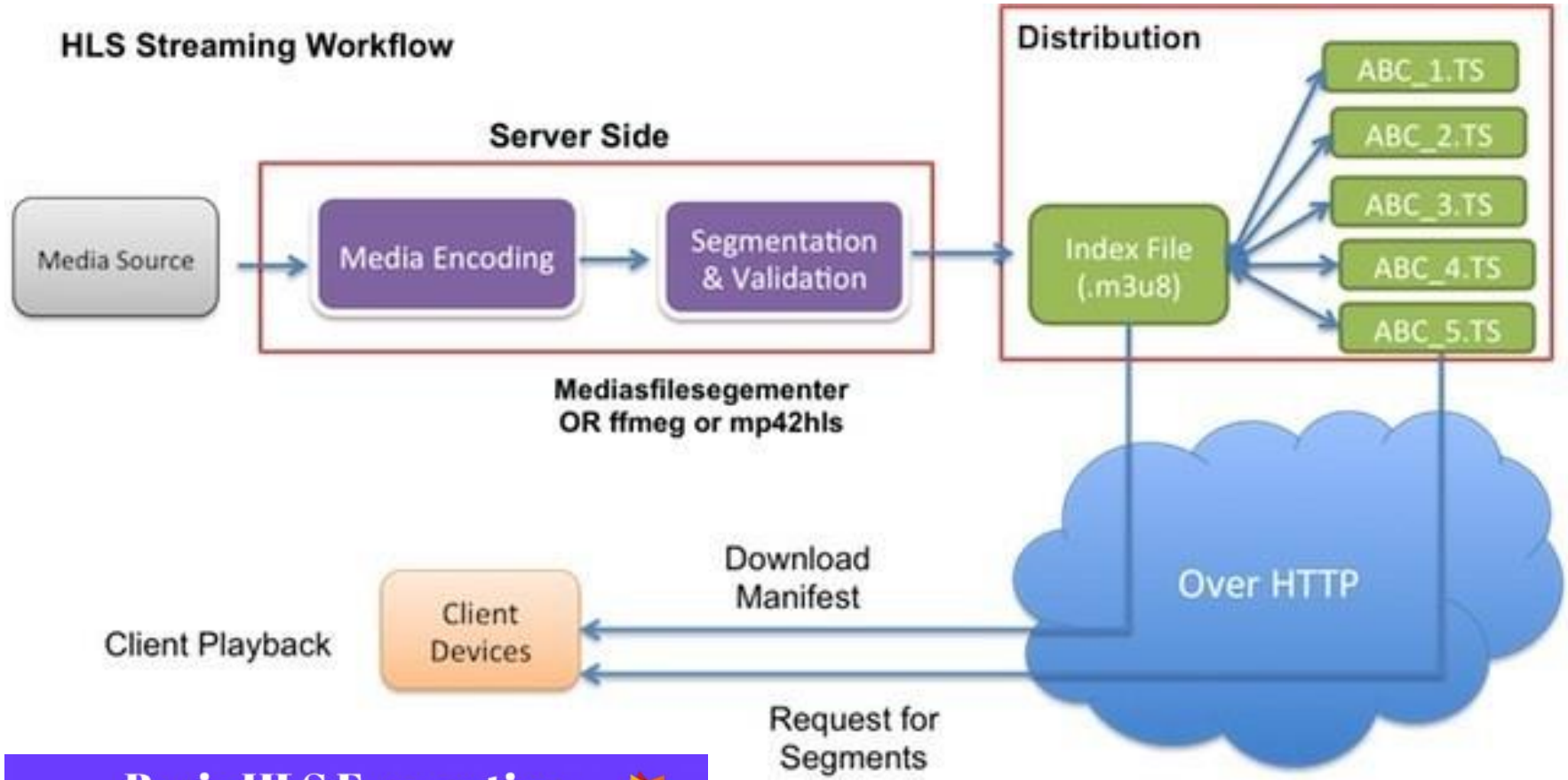
- Feasibility of Fog Computing Deployment based on Docker **Containerization** over RaspberryPi … ICDCN, ACM, 2017
- Virtualization on Internet of Things **Edge** Devices With **Container** Technologies: A Performance Evaluation … IEEE Access, 2017
- **Container**-as-a-Service at the **Edge**: Trade-off between Energy Efficiency and Service Availability at Fog Nano Data Centers … IEEE Wireless Communications, 2017
- **Edge** Computing Embedded Platform with **Container** Migration … IEEE SmartWorld, 2017
- Evaluating Performance of **Containerized** IoT Services for Clustered Devices at the Network **Edge** … IEEE Internet of Things Journal, 2017
- VIoLET: A Large-scale **Virtual** Environment for **Internet of Things** … CoRR, arXiv, 2018
- Video Processing on the **Edge** for Multimedia IoT Systems … CoRR, arXiv, 2018

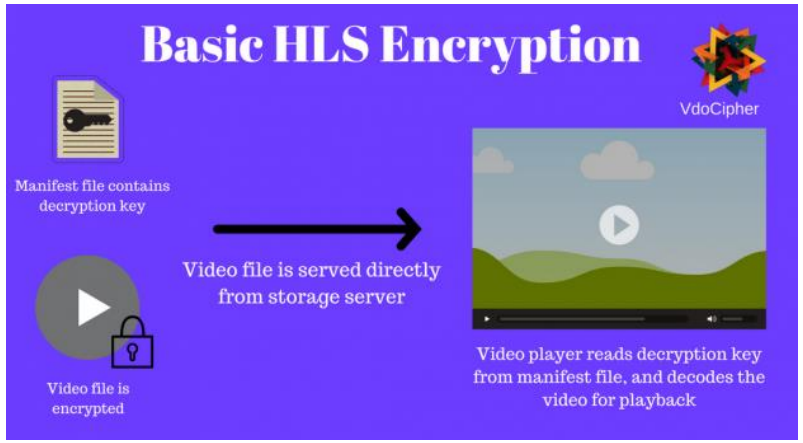
Live Streaming Technology

- Streaming protocols
  - RTSP: IP camera
  - RTMP: Encode and publish to streaming platform
- HTTP-based streaming protocols

	HLS	MPEG-DASH	HTTP-FLV
Open Source Client	hls.js	dash.js	flv.js
Open Source Server	nginx-rtmp-module	nginx-rtmp-module	nginx-http-flv-module
Support iOS Safari	Yes (natively)	No	No
Example	Twitch, LINE TV	Netflix, YouTube Live	Douyu TV, bilibili

HTTP Live Streaming

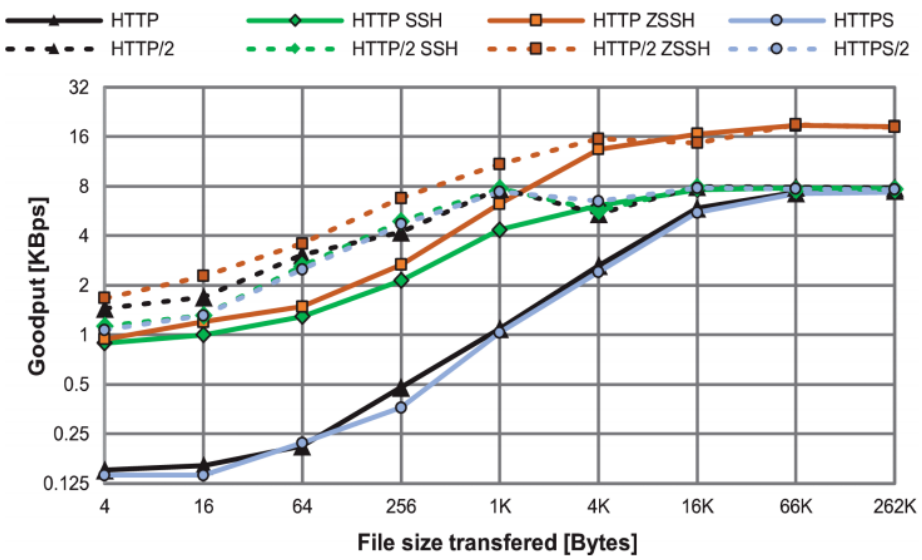




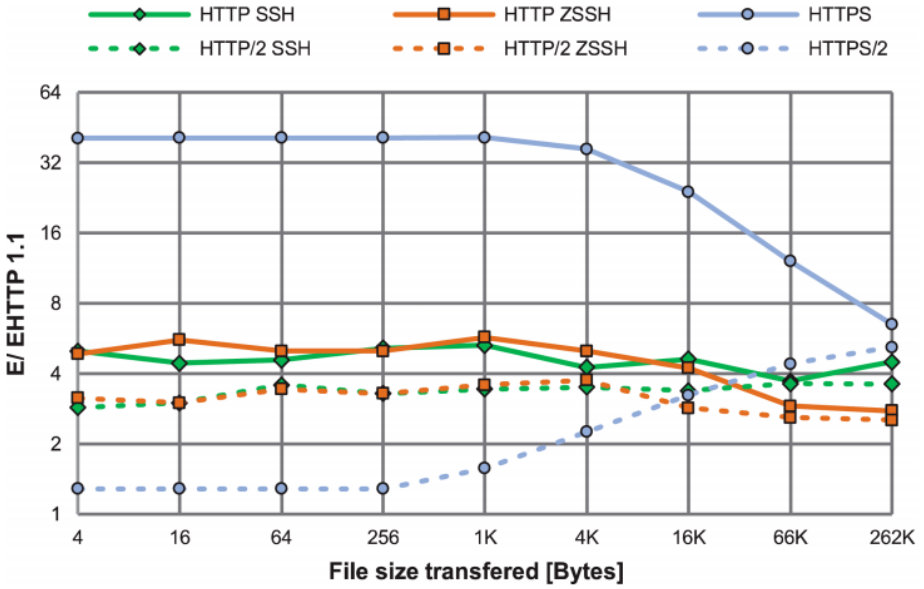
http://www.mediaentertainmentinfo.com/2016/06/6-me-technical-how-to-setup-single-bitrate-hls-streaming.html/
https://www.vdocipher.com/blog/2017/08/hls-streaming-hls-encryption-secure-hls-drm/

SSH Tunnel in IoT

- IoTsafe, Decoupling Security From Applications for a Safer IoT … IEEE Access, 2019
- HTTP over TLS vs. HTTP over SSH tunnel (with ZLIB compression)
  - performed over 802.15.4 LoWPAN on top of two Raspberry Pis



(a) Goodput

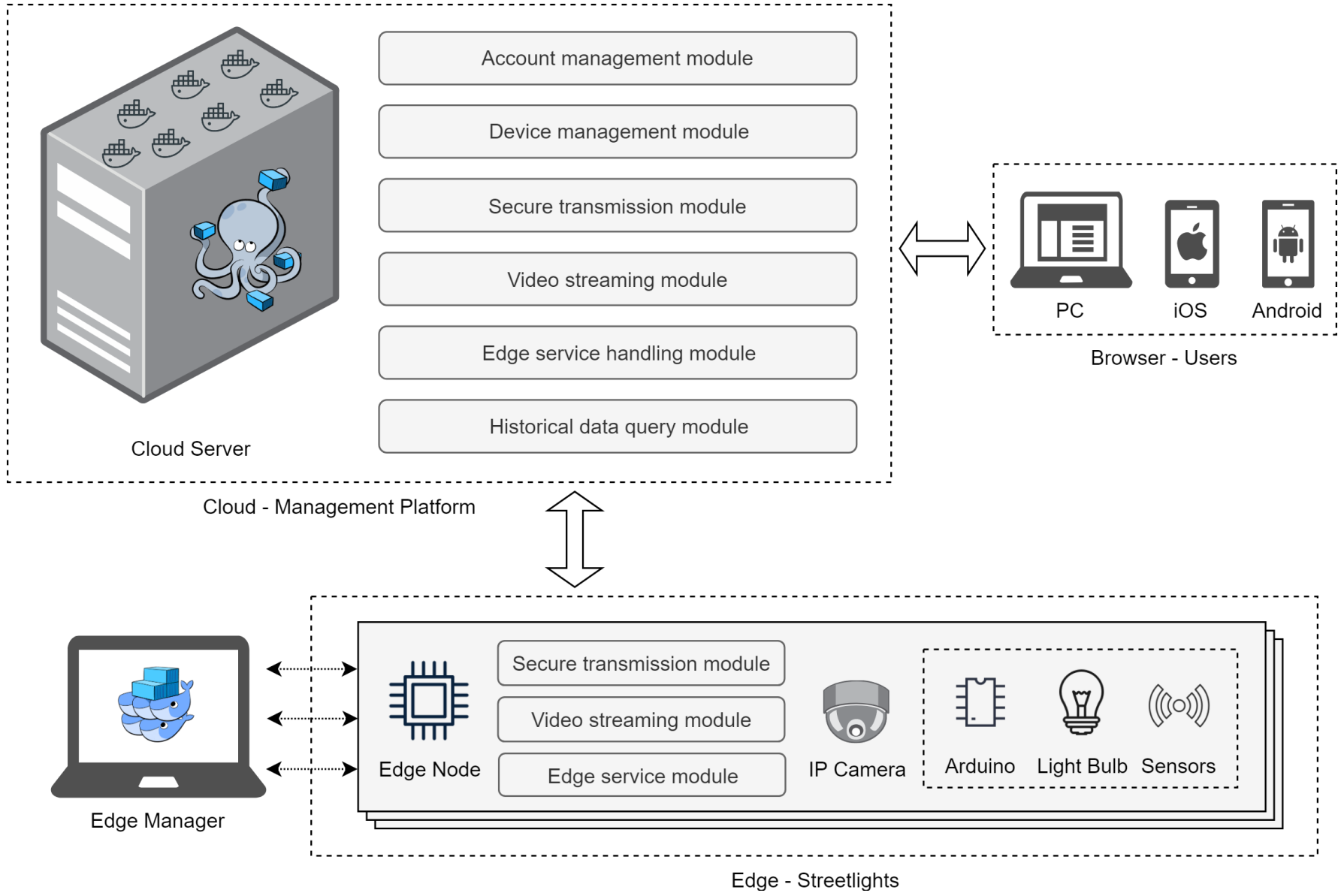


(b) CPU energy consumption

Outline

- Introduction
- Background and Related works
- System Design
- Experiment and Verification
- Conclusion and Future works

System Architecture



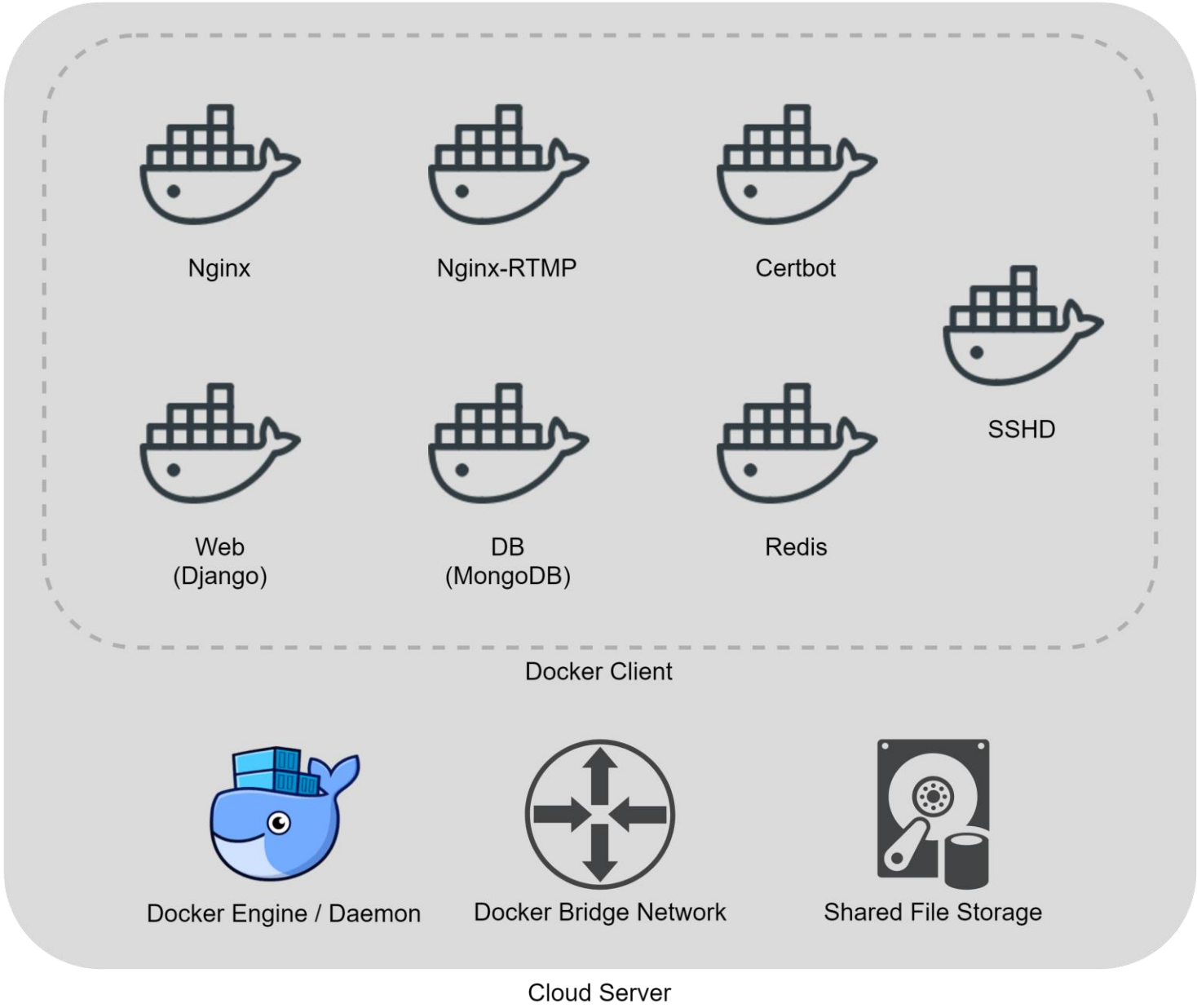
Outline

- Introduction
- Background and Related works
- System Design
  - Cloud Management System
  - Edge Services Orchestrator
- Experiment and Verification
- Conclusion and Future works

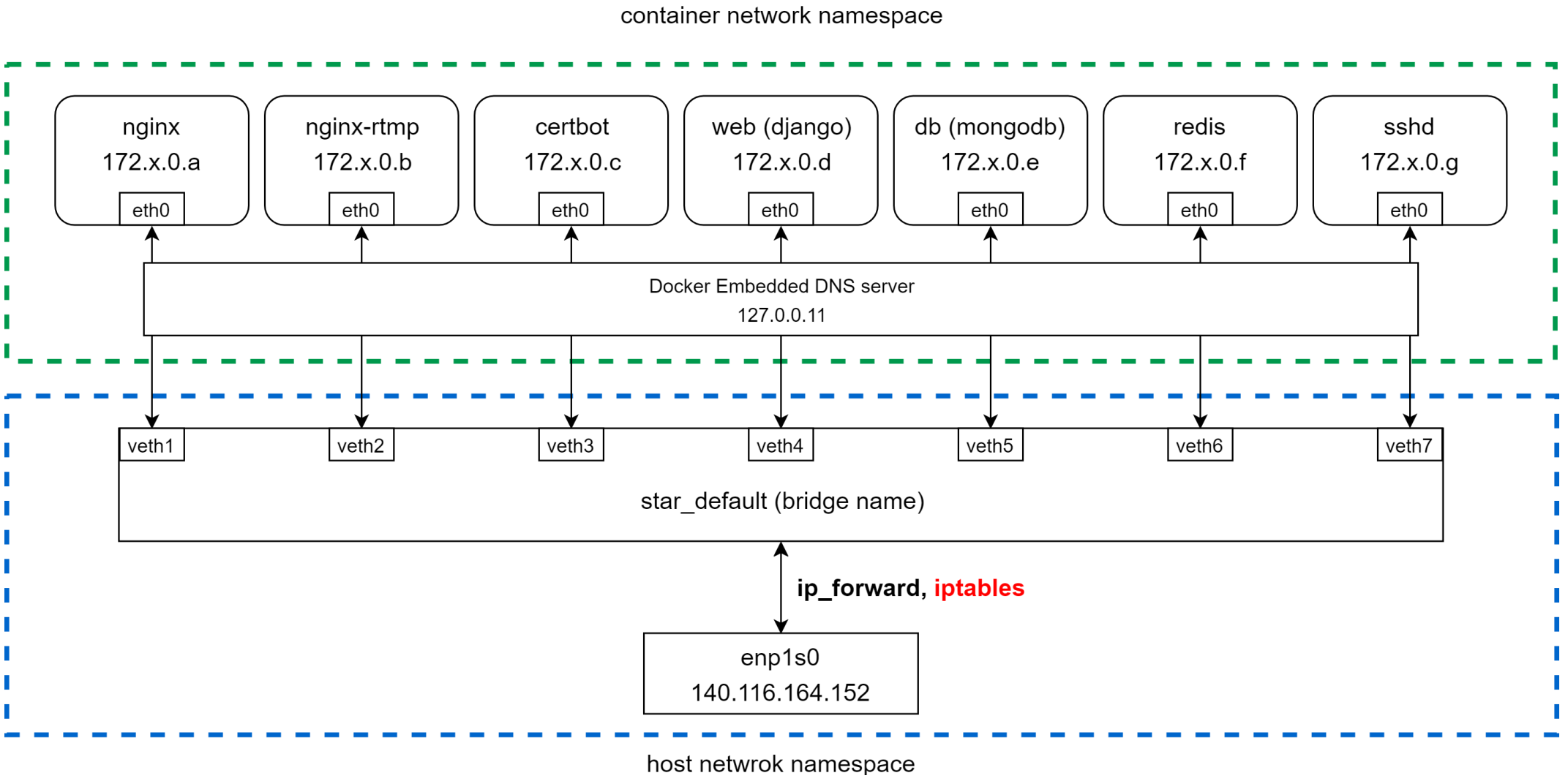
Cloud Management Platform

- Cloud Server Architecture
  - Overview
  - Network
  - Docker images, containers
- Cloud Server Modules
  - Account management
  - Device management
  - Secure transmission
  - Video streaming
  - Edge service handling
  - Historical data query

Cloud Server - Overview



Cloud Server – Docker Bridge Network



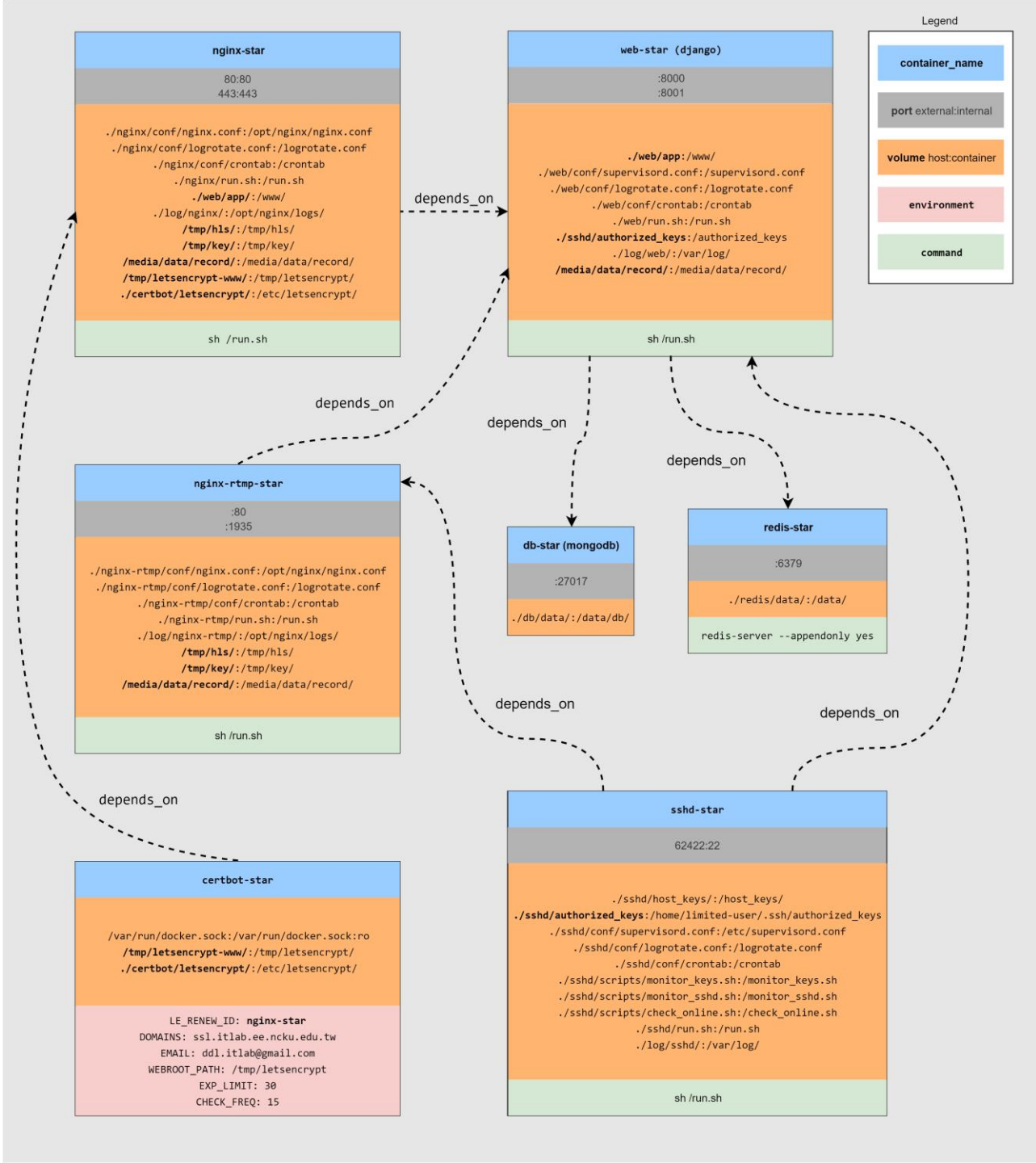
Cloud Server - Docker Images

Image / Info	Base Image	Size
Nginx	alpine:3.9	23MB
Nginx-RTMP	alpine:3.9	112MB
Web (Django)	python:3.7.3-alpine3.9	280MB
MongoDB	mvertes/alpine-mongo:4.0.61	123MB
Redis	redis:5.0.5-alpine3.9	50.9MB
SSHD	alpine:3.9	57MB
Certbot	docker:18.09.6	256MB

Cloud Server - Docker Containers

Ports exposed:

- 80
  (HTTPS certificate renewal)
- 443
  (HTTPS requests)
- 62422
  (SSH Tunnel)



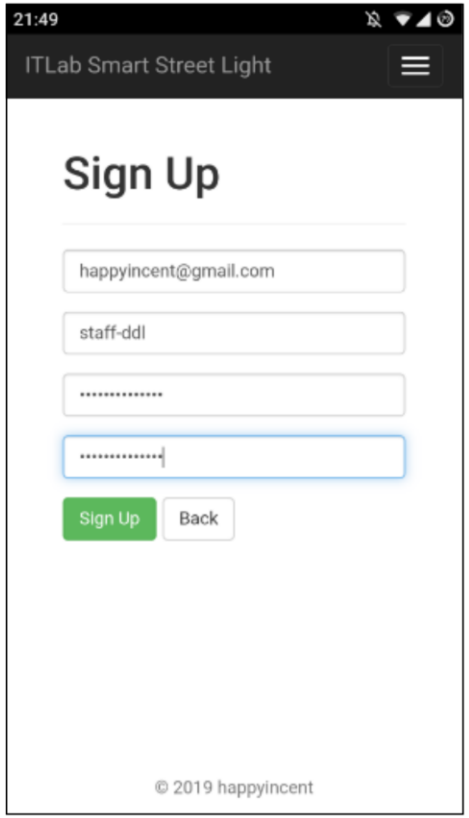
Cloud Management Platform

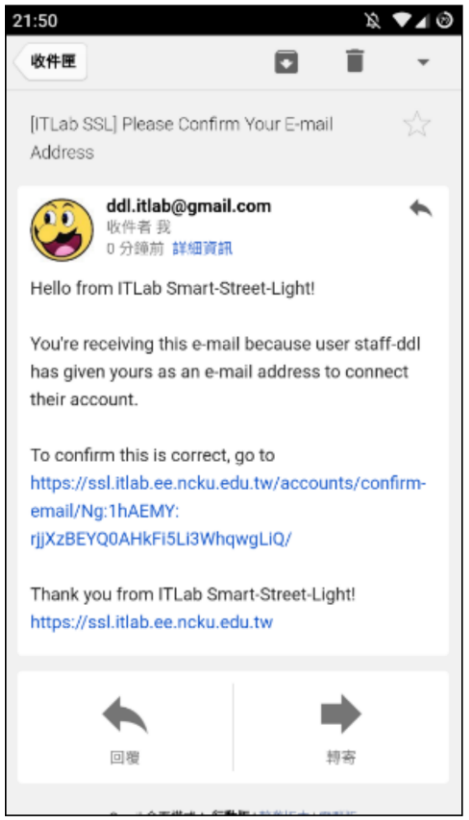
- Cloud Server Architecture
  - Overview
  - Network
  - Docker images, containers
- Cloud Server Modules
  - Account management
  - Device management
  - Secure transmission
  - Video streaming
  - Edge service handling
  - Historical data query

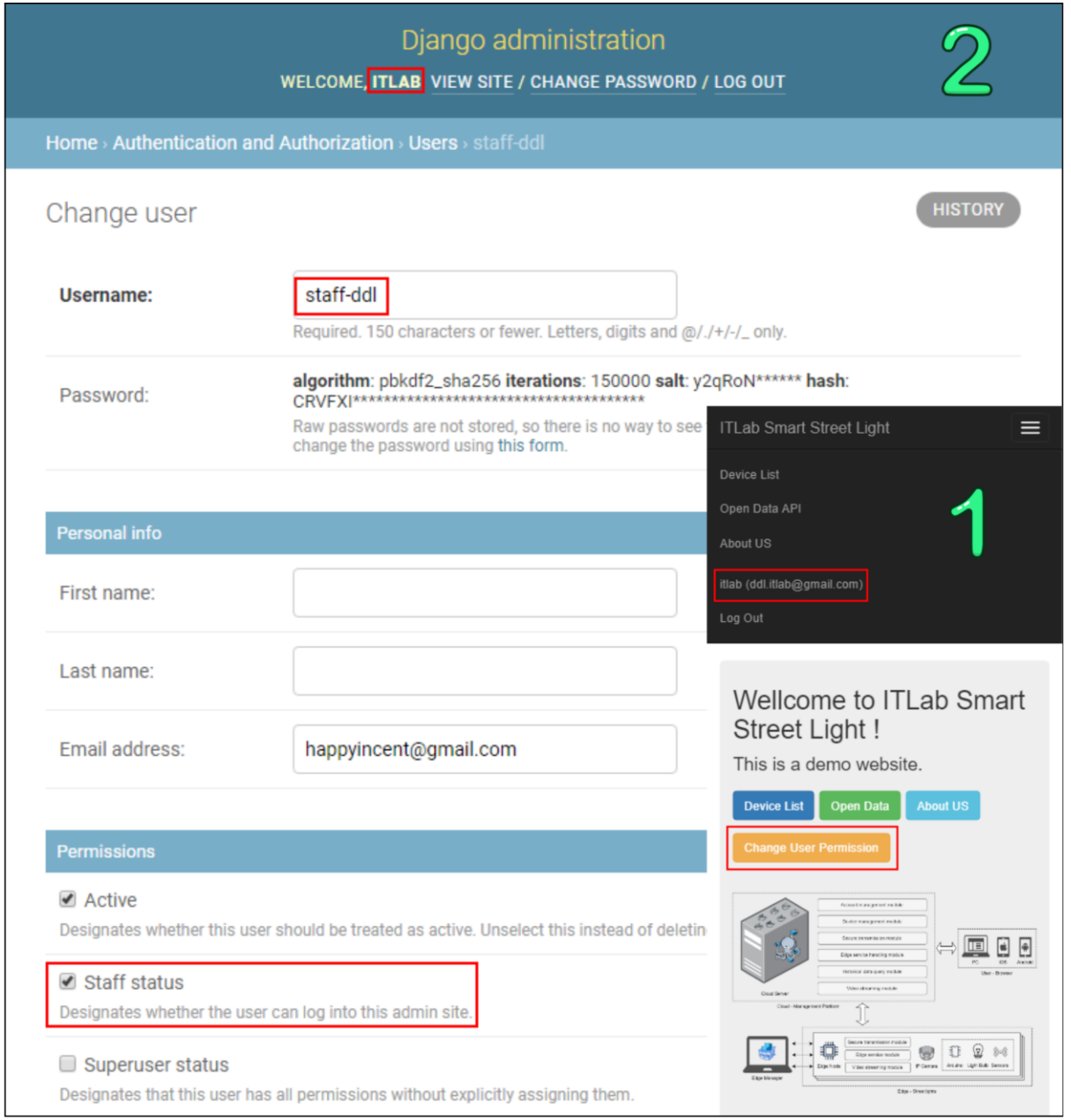
Account Management

Type \ Permission	Get Data/Streaming	Config LED (Timetable)	Setup Device Configuration	Edit User Permission
Superuser	√	√	√	√
Staff	√	√	√ (creator)	
Email Verified	√			
Email not Verified				

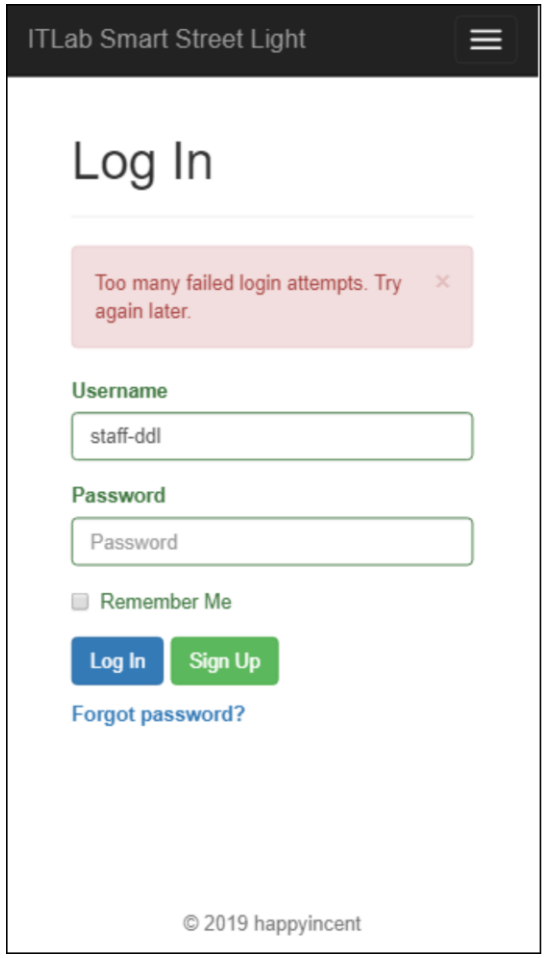
Account Management

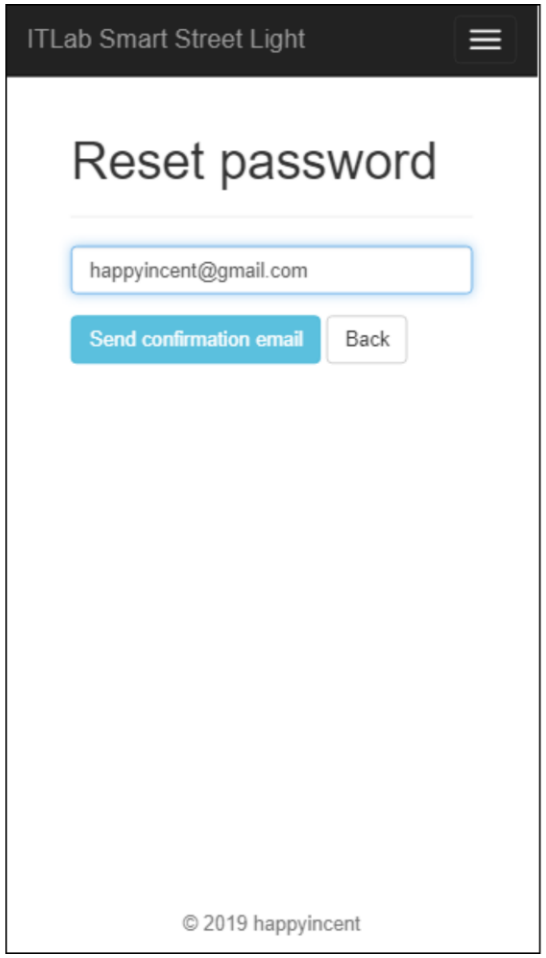


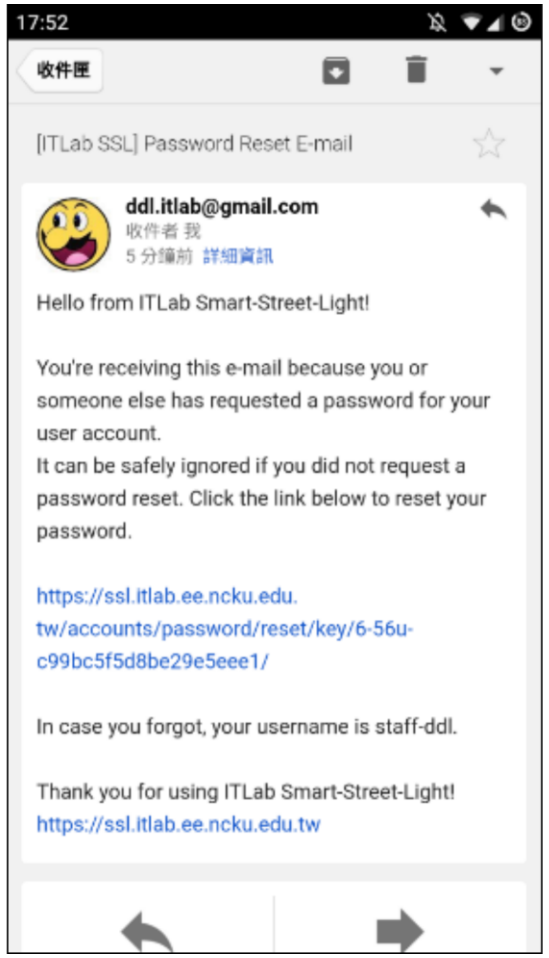


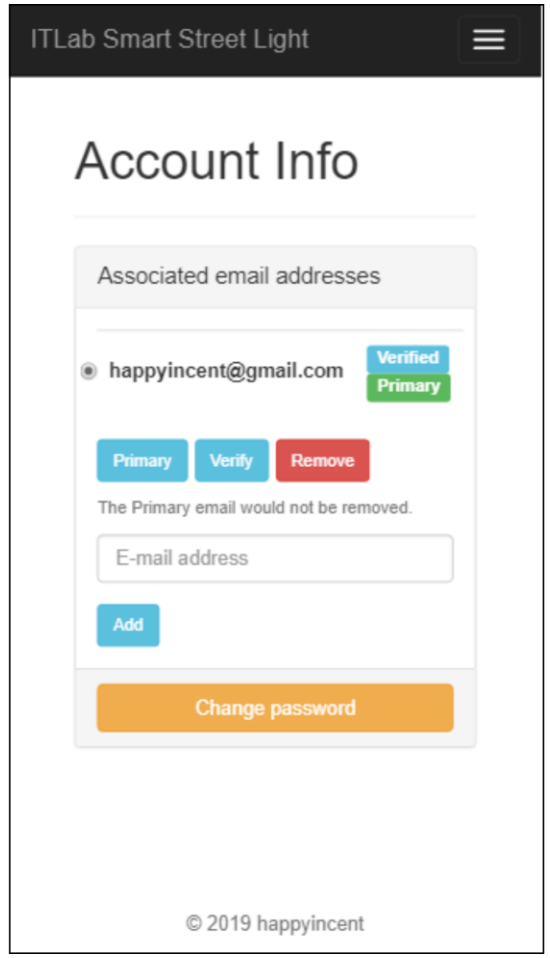


Account Management





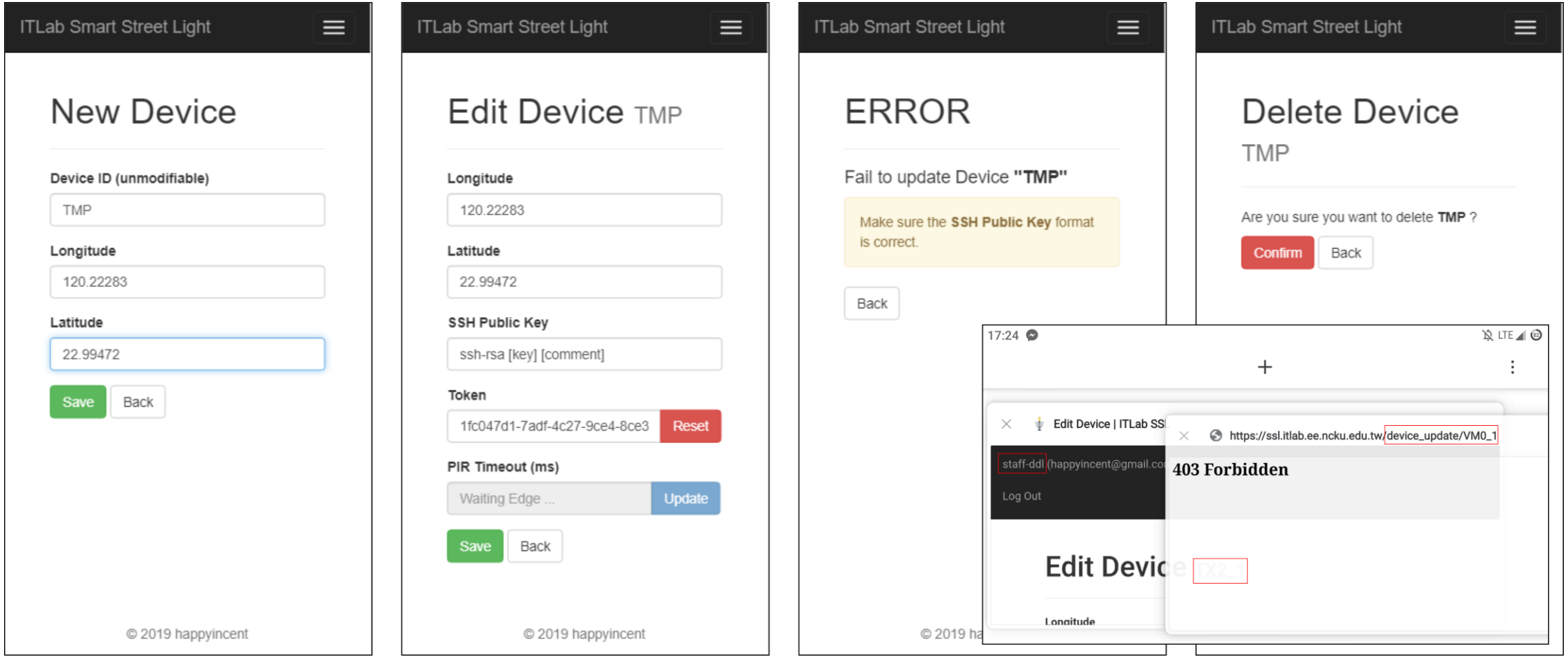




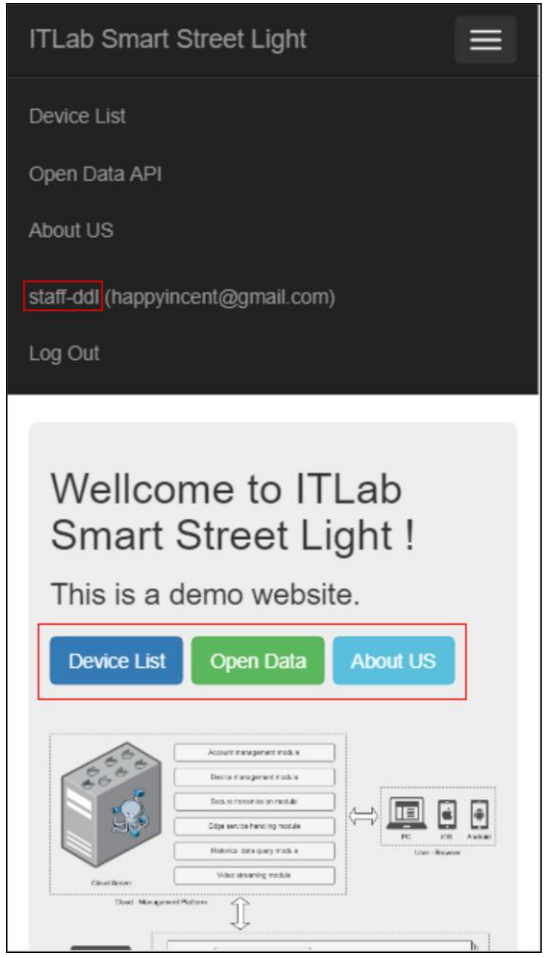
Device Management

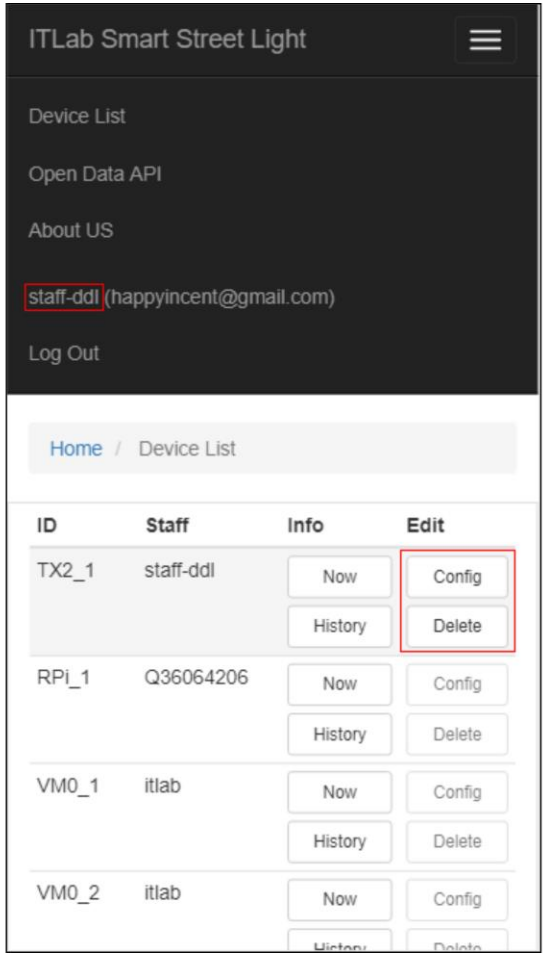
- Maintain Device Collection (stored in MongoDB)
  - User ID
  - Device ID
  - Longitude / Latitude
  - Token
  - SSH Public Key
  - LED / PIR schedule
- Process of CRUD (with permission check)
  - Generate and store token into Redis
    (create device, reset token)
  - Drop connections
    (update public key, reset token, delete device)

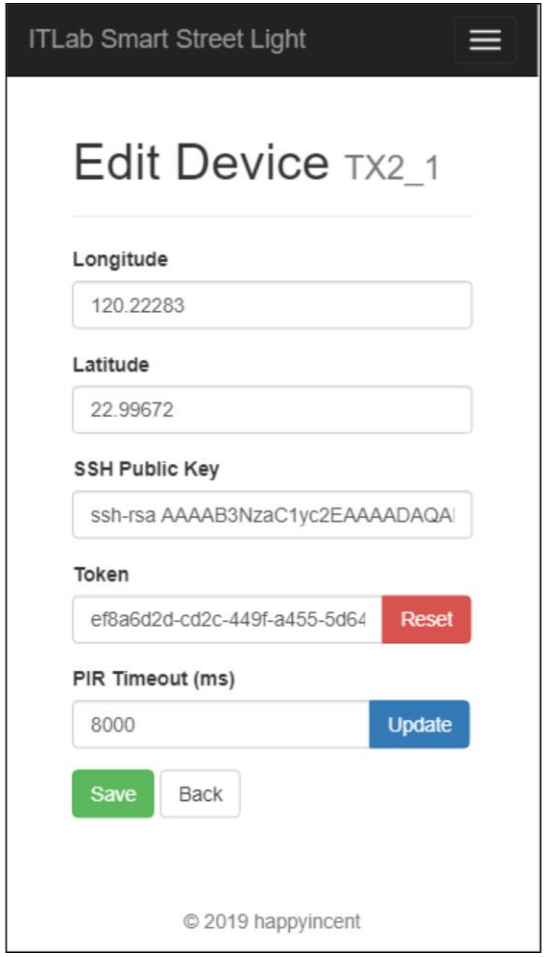
Device Management

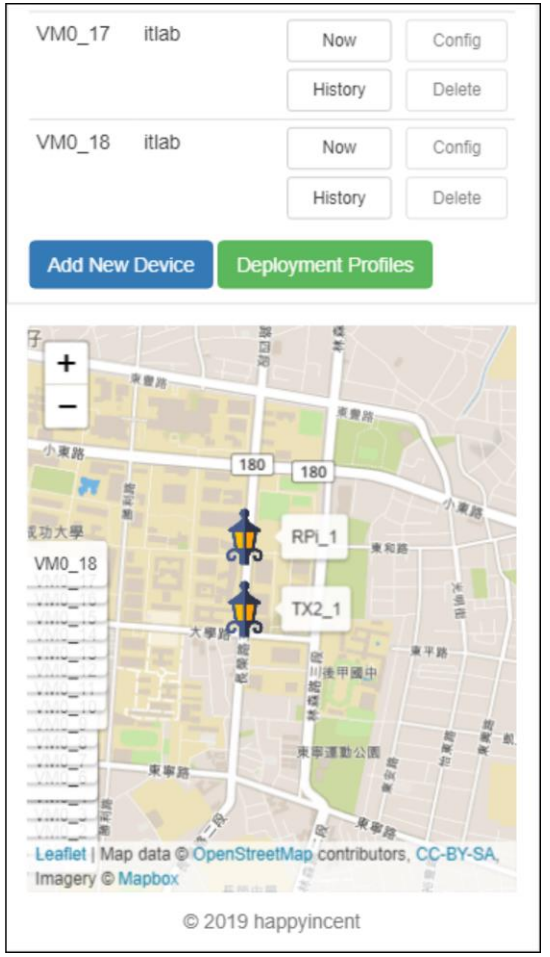


Device Management

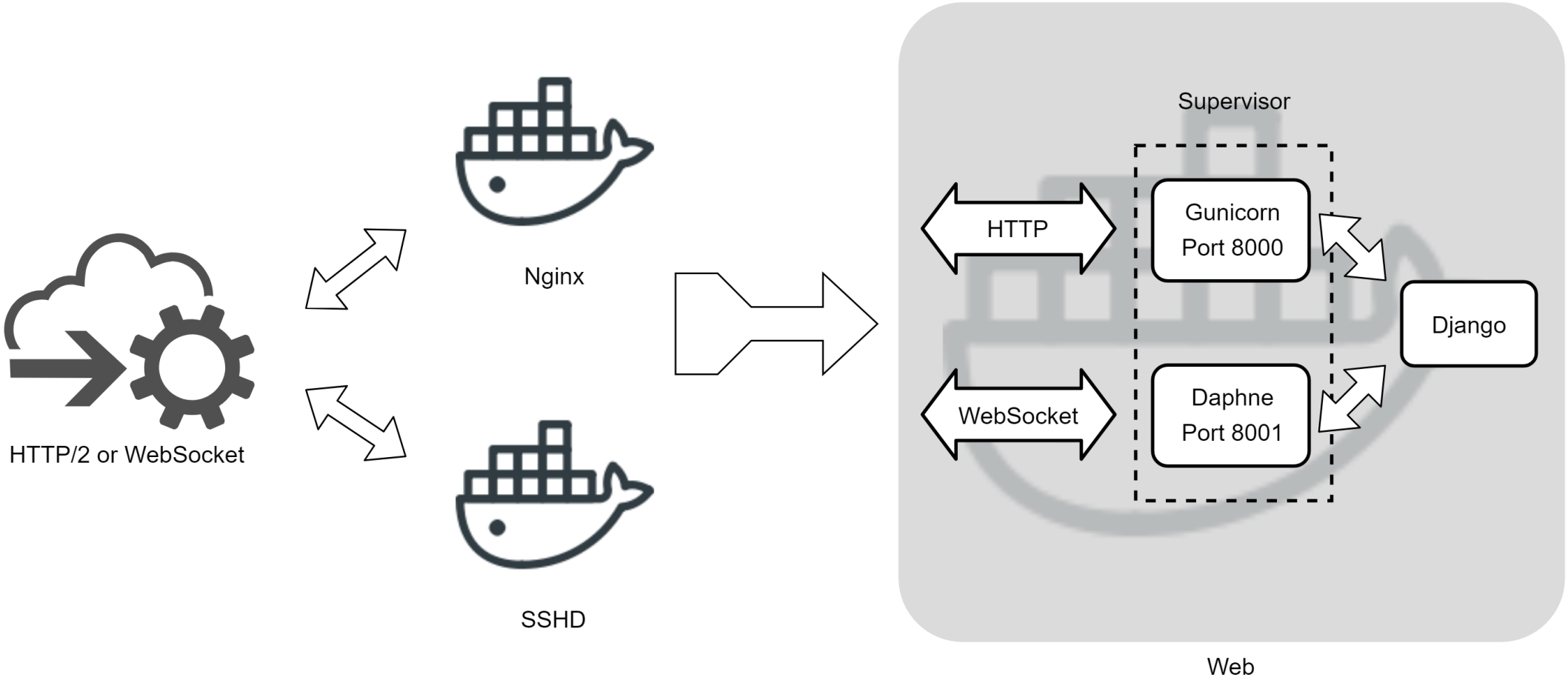




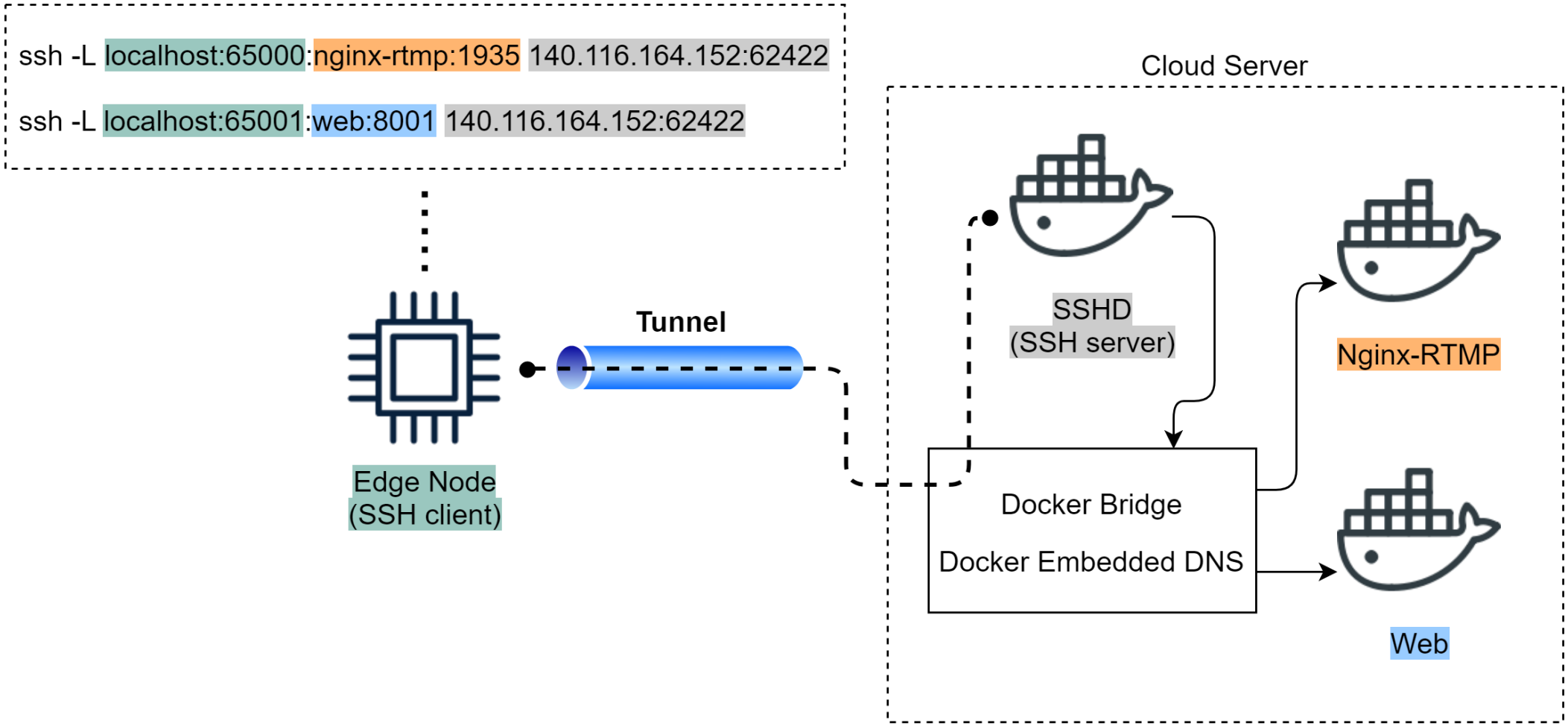




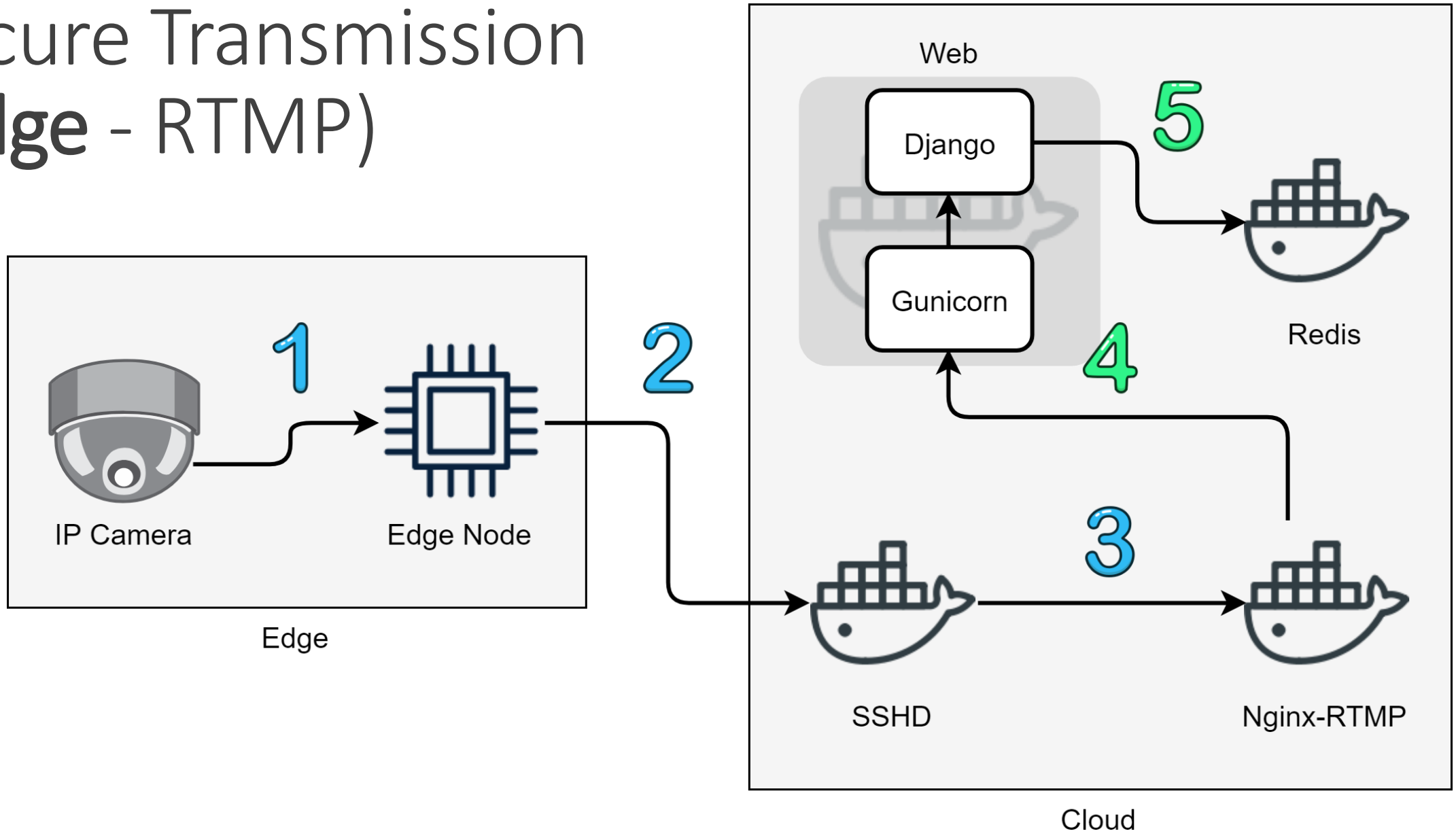
Web-based Requests Workflow



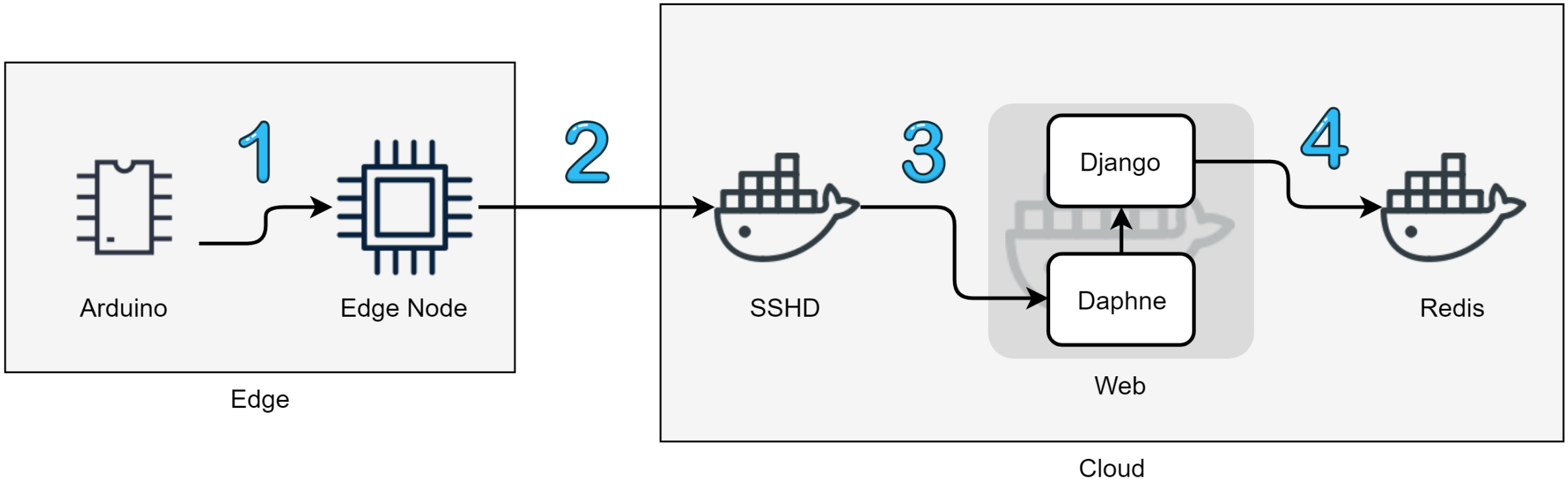
Secure Transmission (**Edge**)



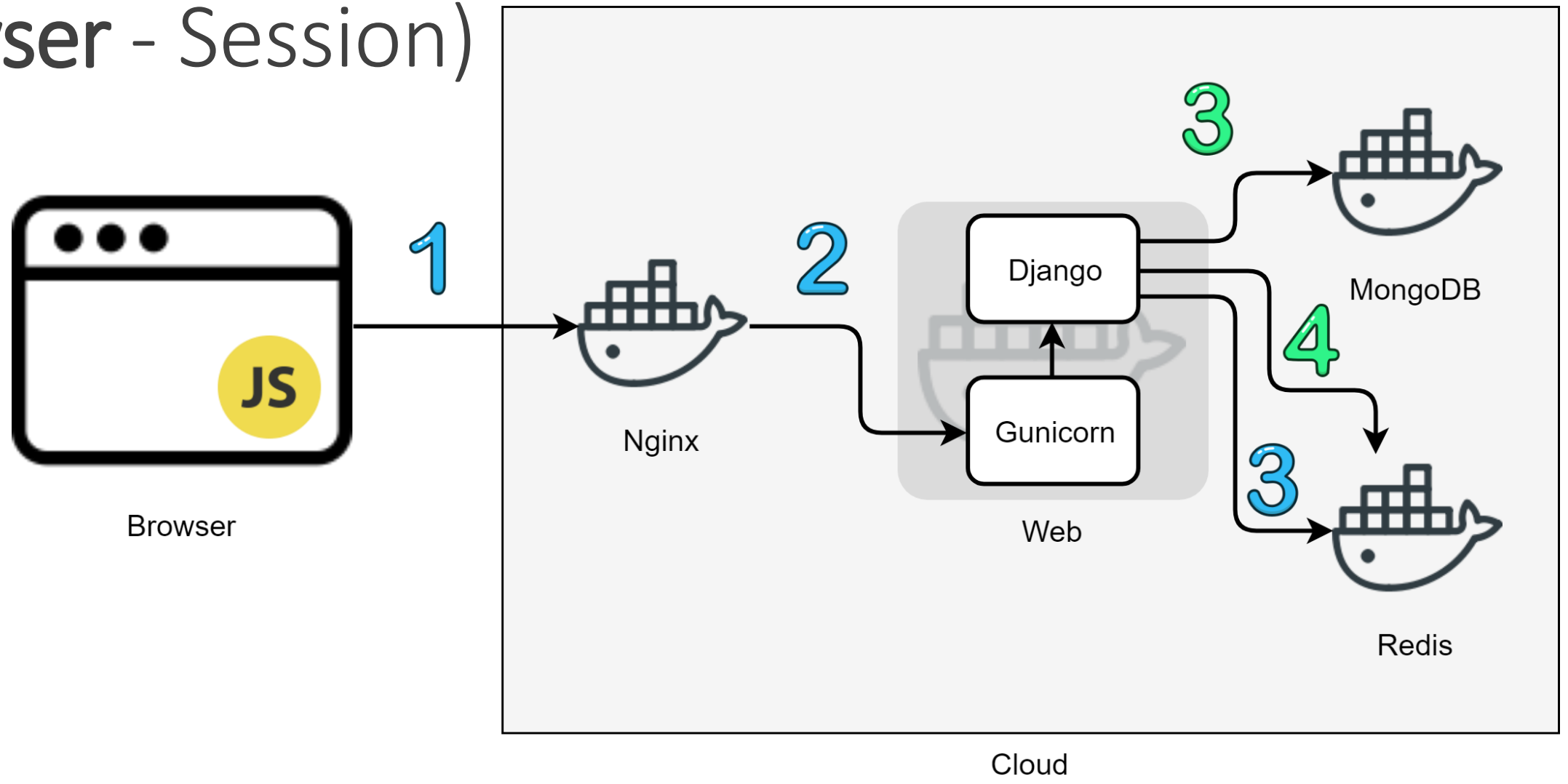
Secure Transmission (**Edge** - RTMP)



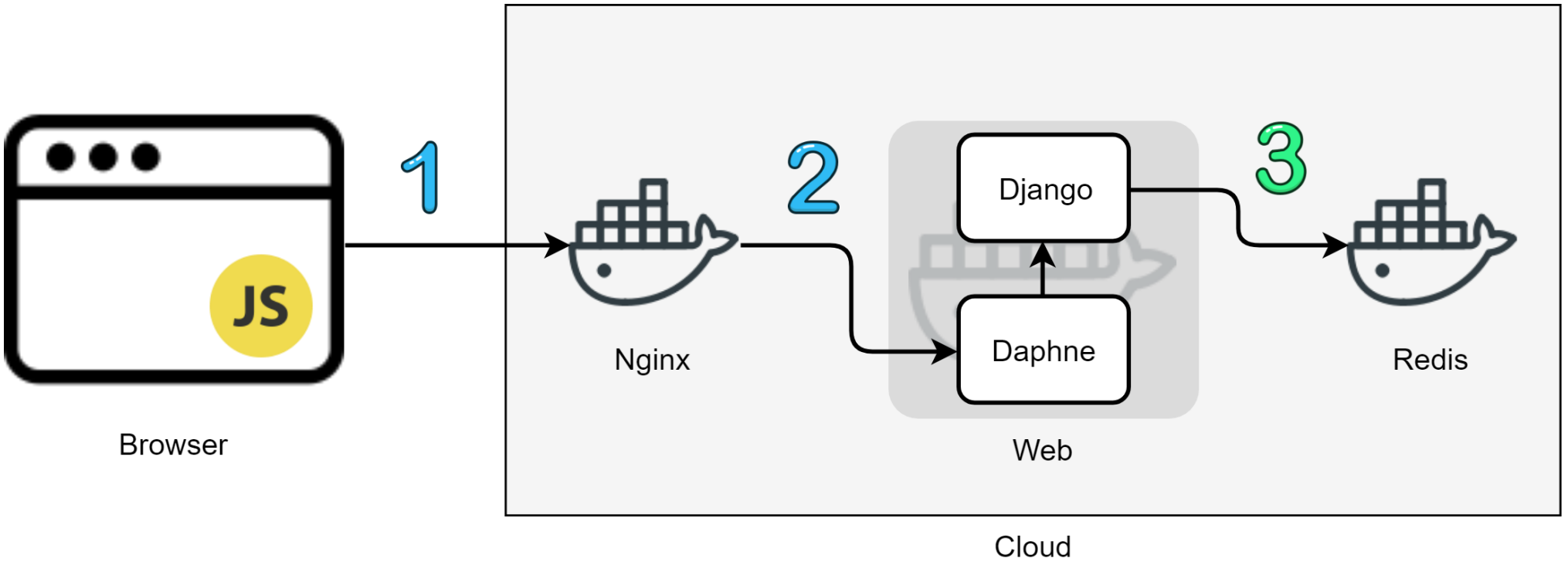
Secure Transmission (**Edge** - WebSocket)



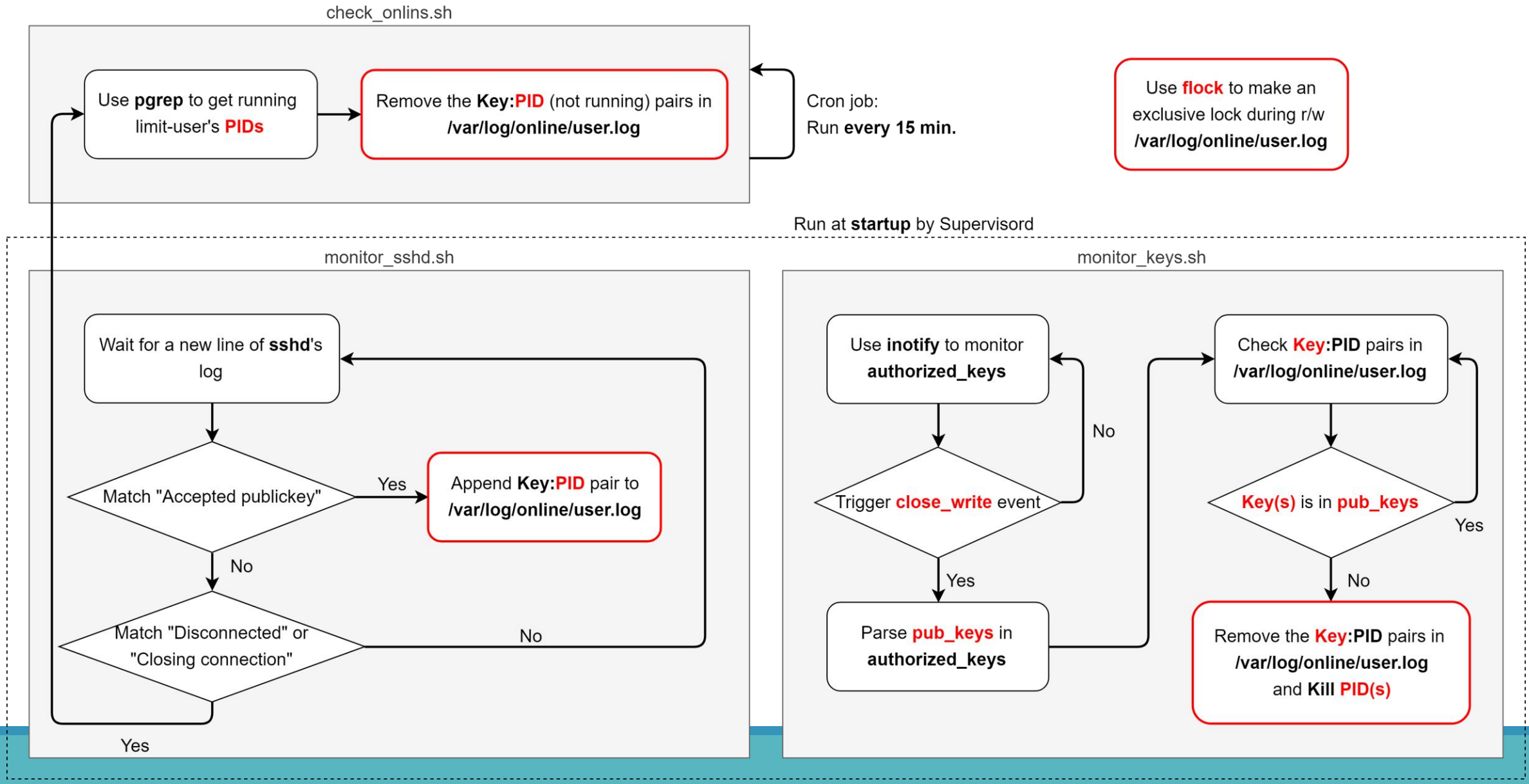
Secure Transmission (**Browser** - Session)



Secure Transmission (**Browser** - WebSocket)



Valid SSH User Checking



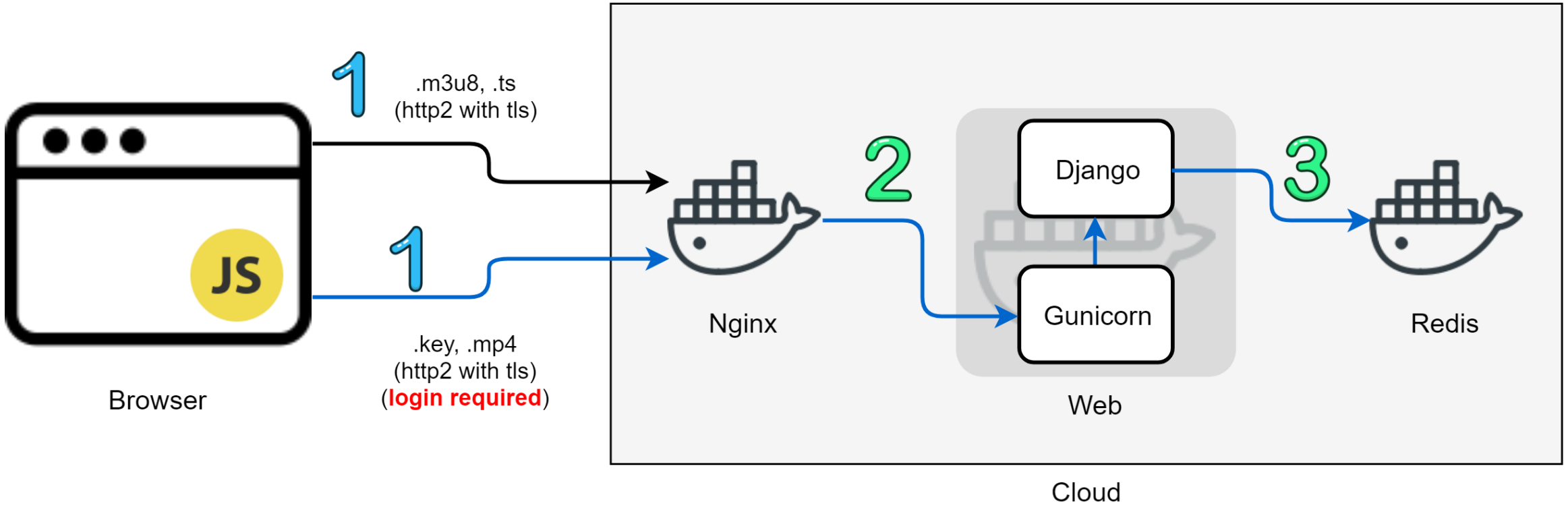
Certbot – HTTPS certificate Renewal

- Let’s Encrypt
  - a free, automated, and open Certificate Authority
- Certbot
  - an extensible client for the Let’s Encrypt CA
- Every 15 days → check if the certificate is expired in 30 days

Video Streaming (Nginx-RTMP)

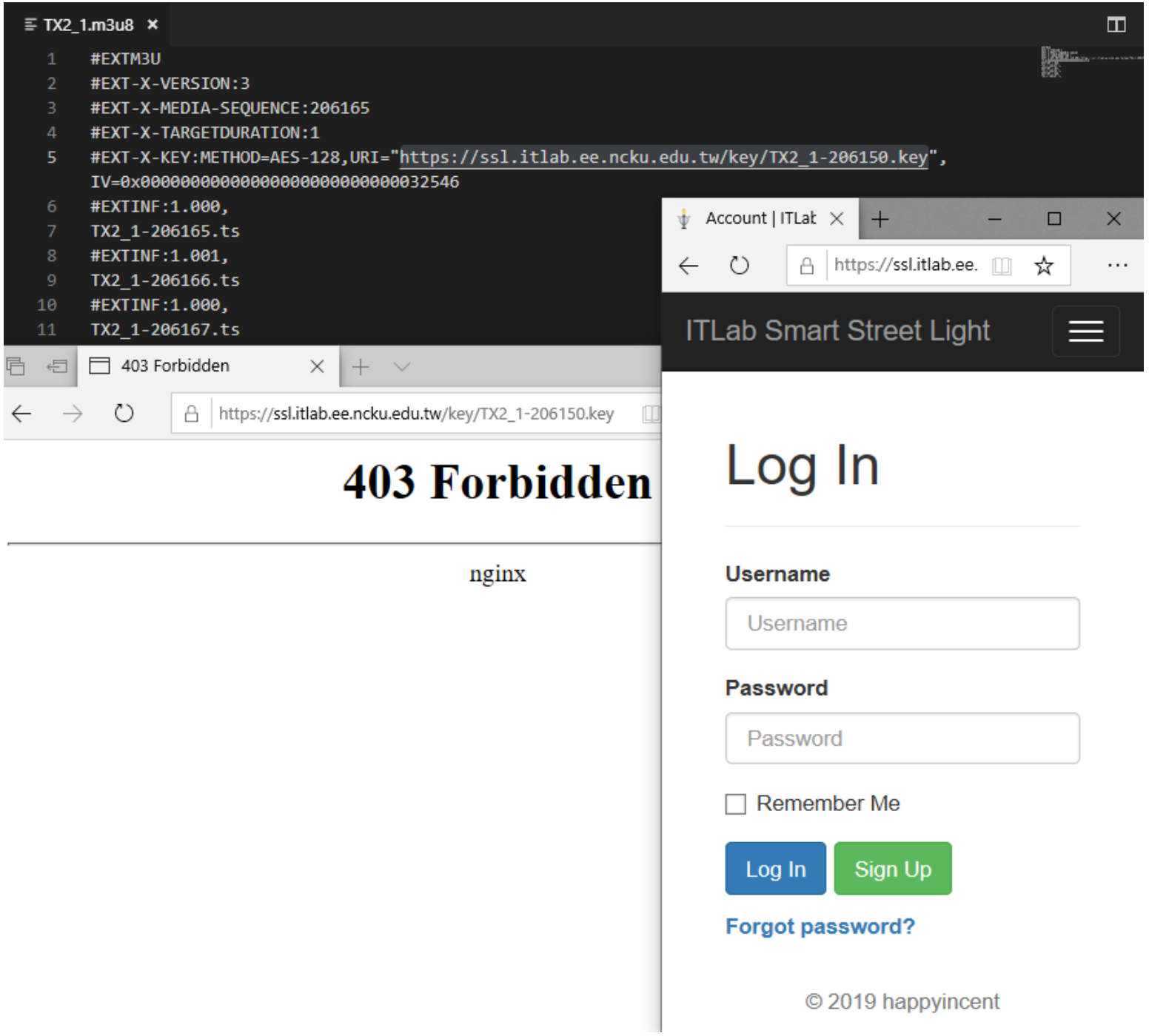
- Listen on the internal port 1935 to receive **RTMP** stream with token check
- Generate HLS files
  - HLS playlist (.m3u8)
  - Encrypted media segment (.ts)
  - AES encryption key (.key)
- Record RTMP stream (FLV) and convert it into MP4 by FFmpeg
  - Maximum length of the historical video: 15 minutes
- Listen on the internal port 80 to receive control messages (sent by Django) to drop the RTMP connection (stream name=device id) if
  - Reset device **token**
  - **Delete** device

Secure Transmission (**Browser** - Video)



Video Streaming

HLS Encryption

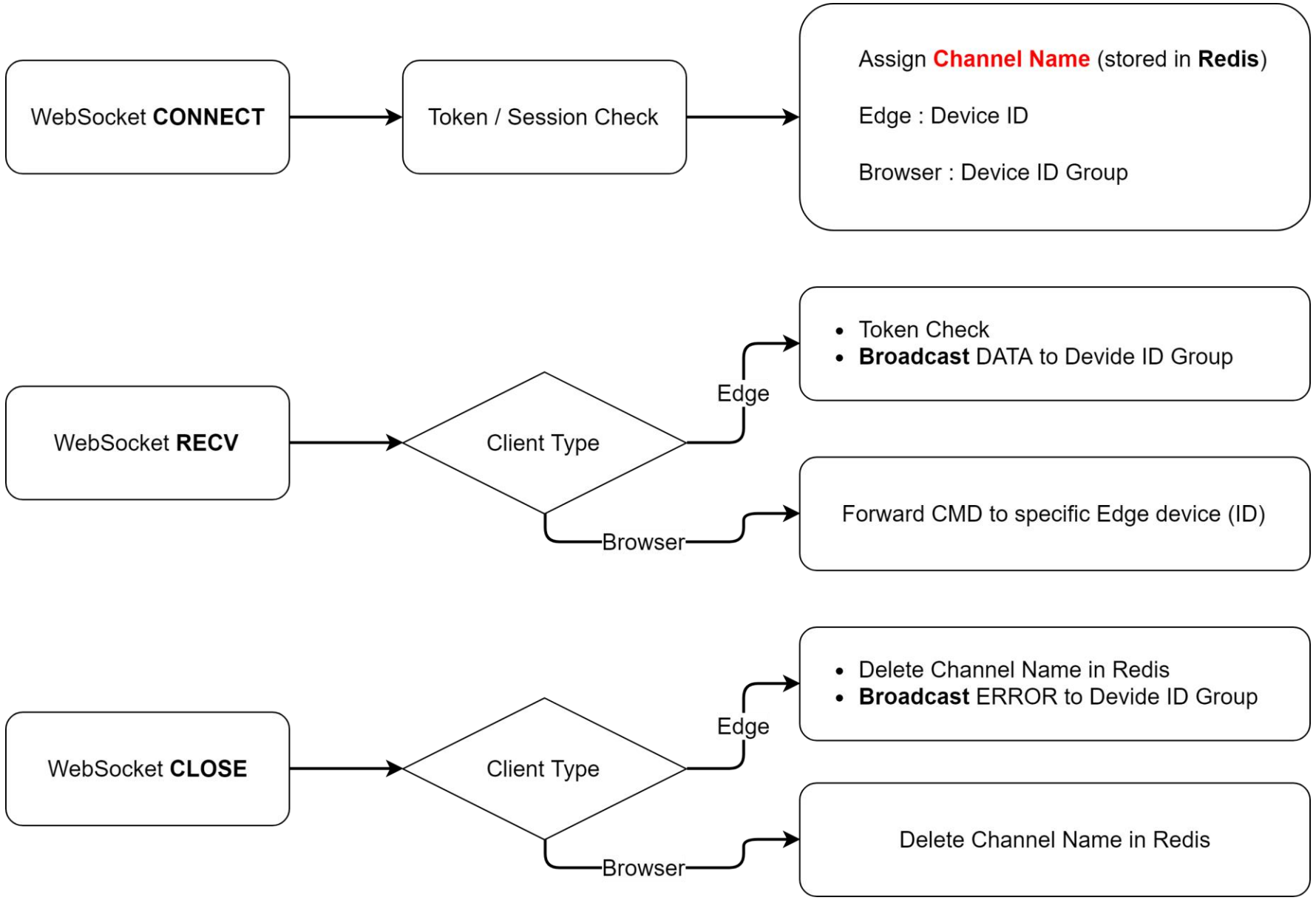


Edge Service Handling (Cloud Module)

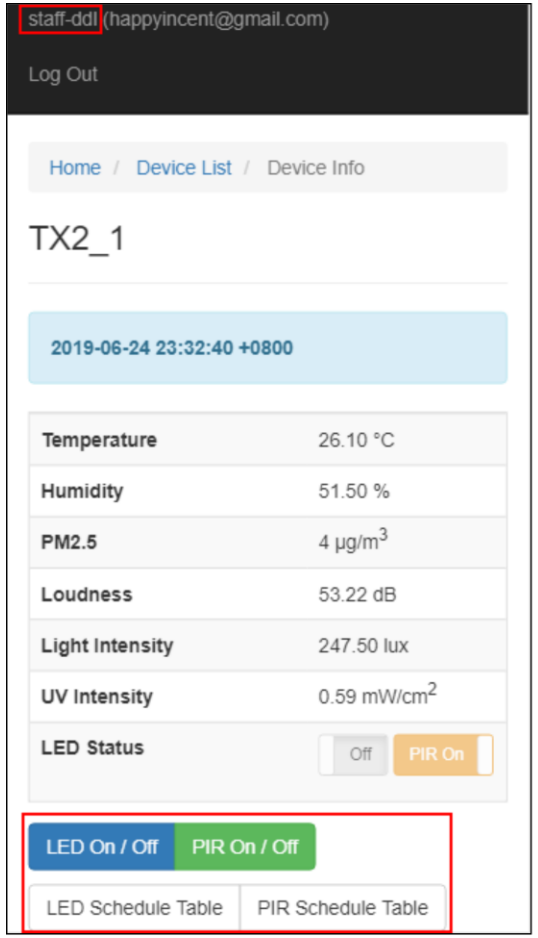
- Listen on the internal port 8001 to receive **WebSocket** connection
  - With token / session check
- Receive information sent **from** Edge and broadcast **to** Browsers (real-time)
- Receive commands sent **from** Browser and forward **to** the particular Edge (real-time)
  - Turn On/Off LED
  - Turn On/Off PIR
  - Change PIR timeout
- Send commands (LED/PIR) **to** the particular Edge every 15 min.
  - According to two timetables stored in MongoDB

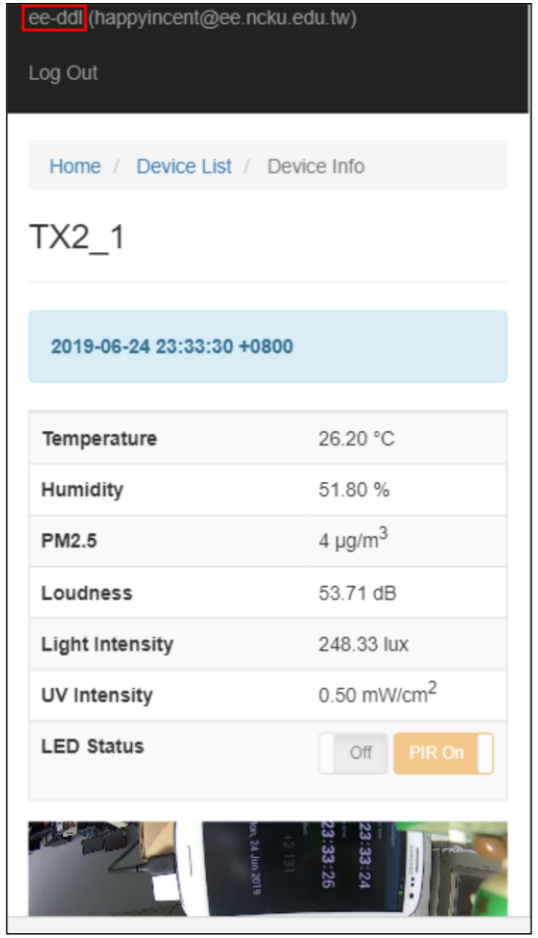
Edge Service Handling
(Cloud Module)

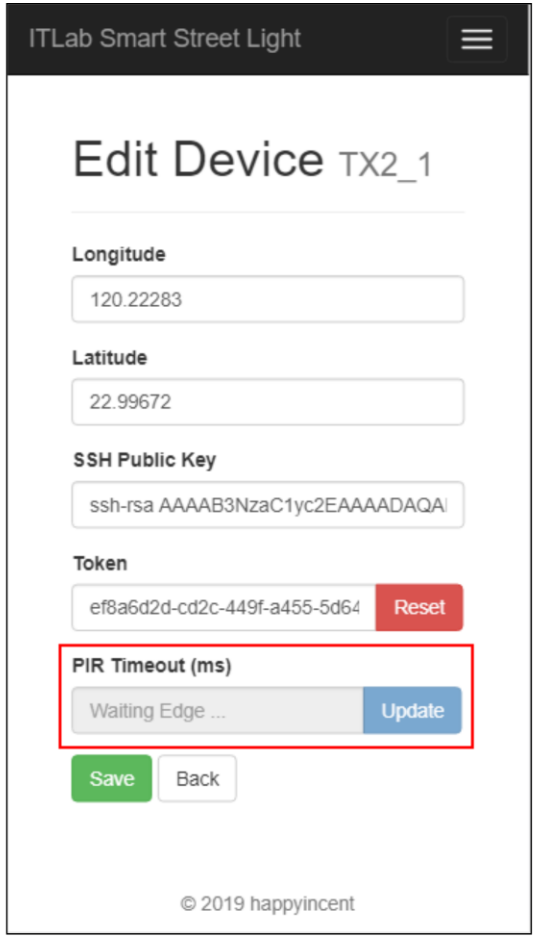
WebSocket

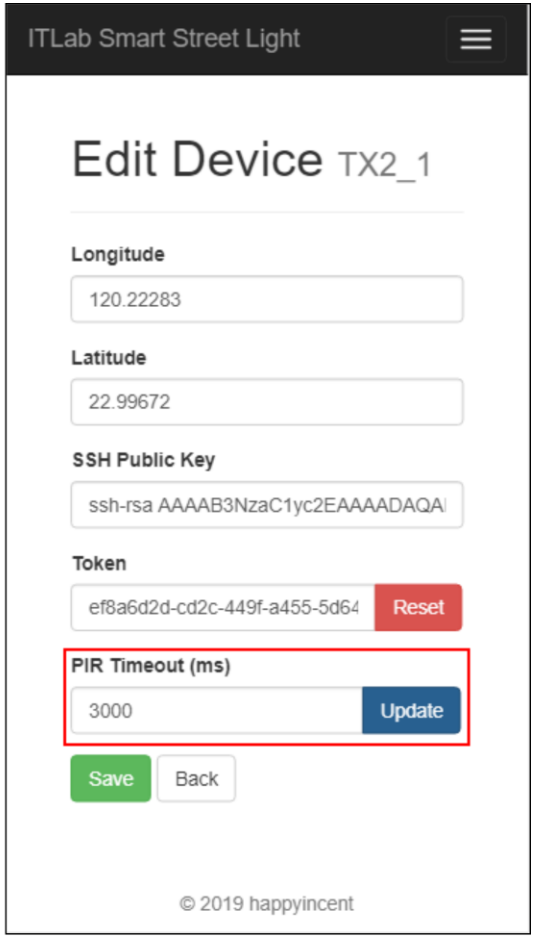


Edge Service Handling (Cloud Module)



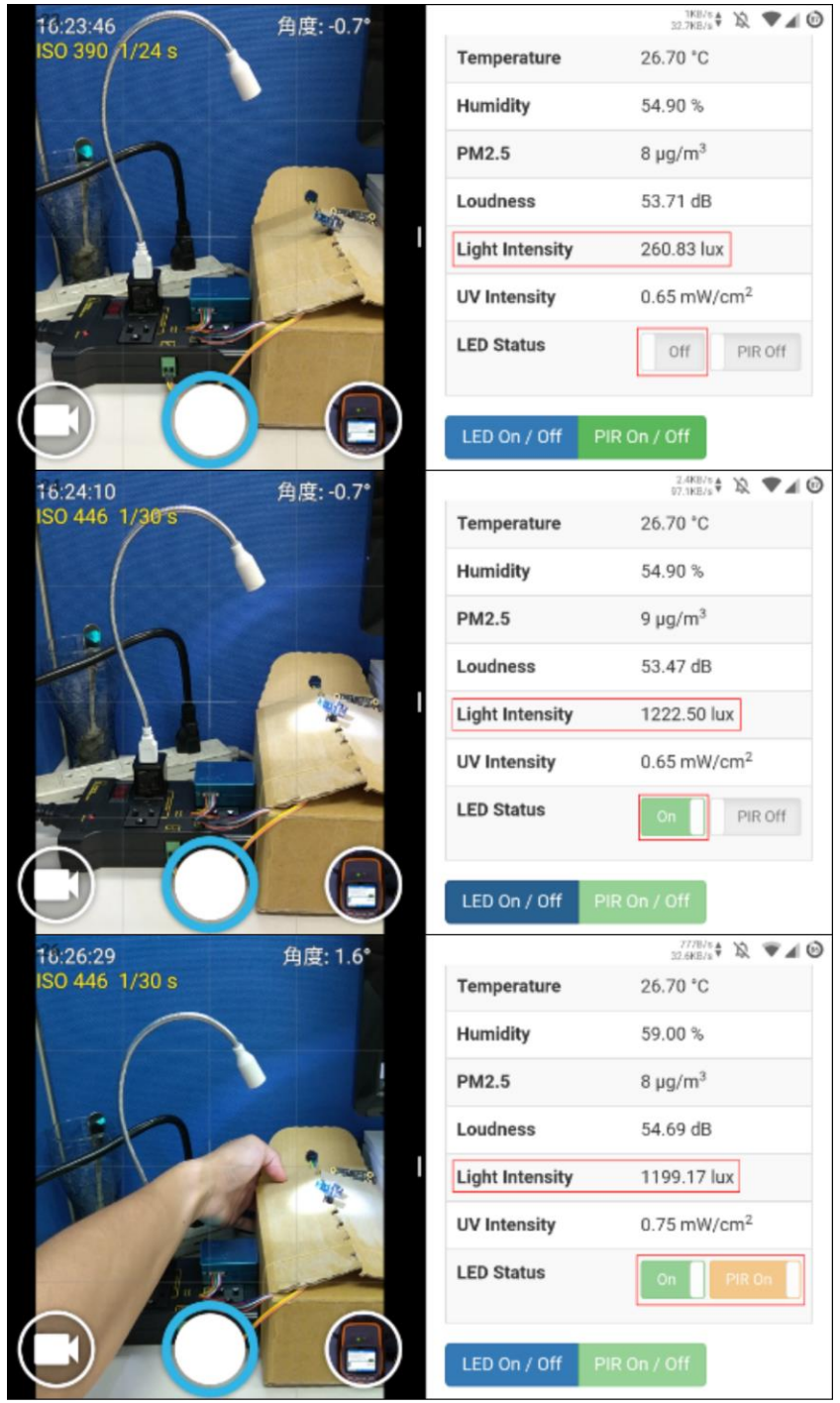






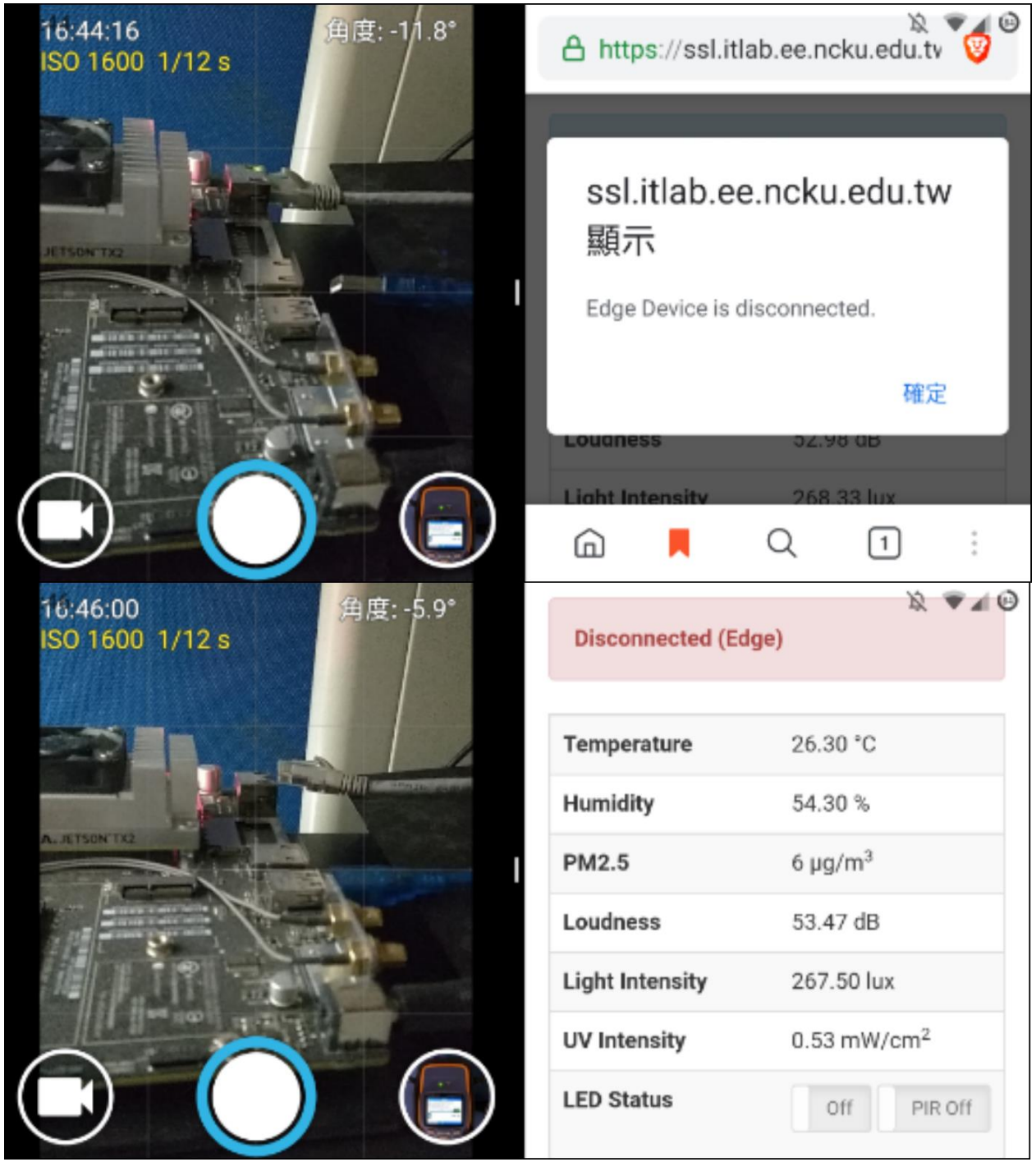
Edge Service Handling
(Cloud Module)

- LED **OFF** & PIR OFF
- LED **ON** & PIR OFF
- LED **ON** & PIR **ON**

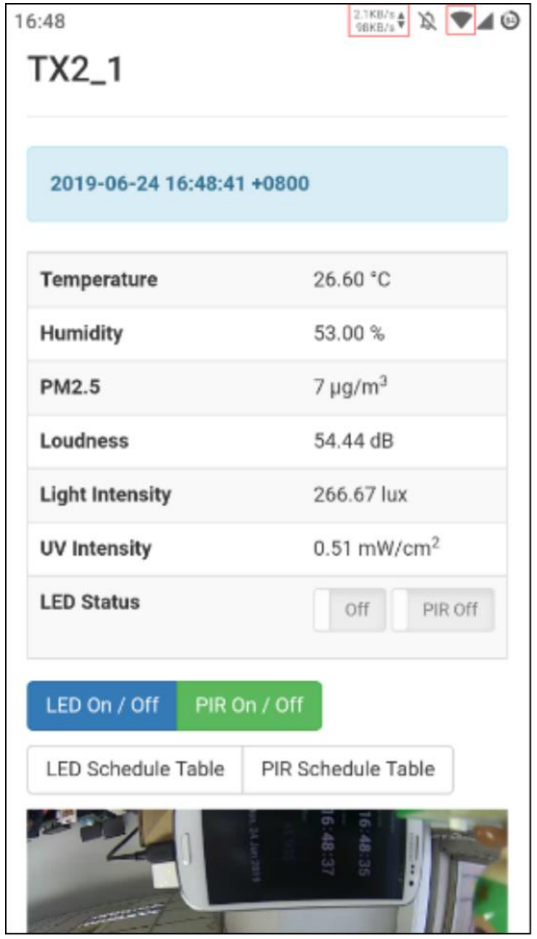


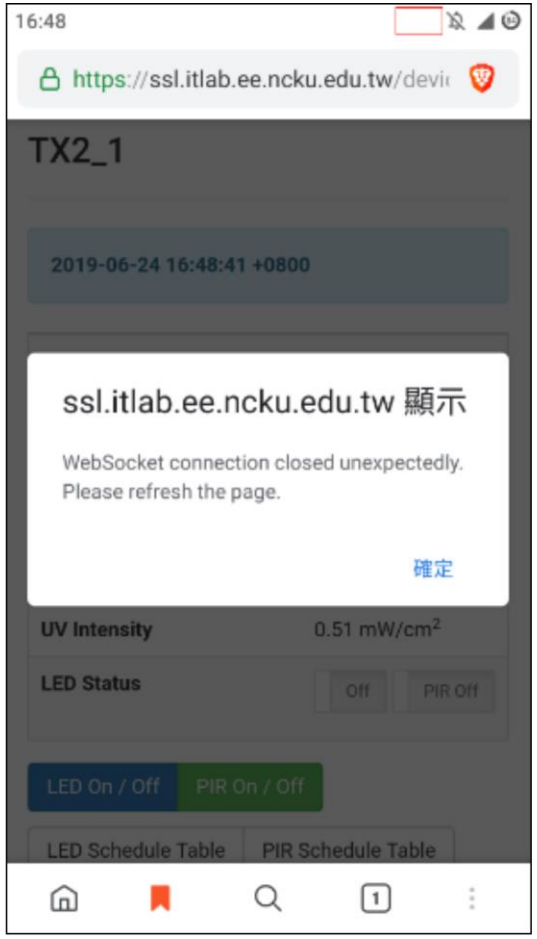
Edge Service Handling
(Cloud Module)

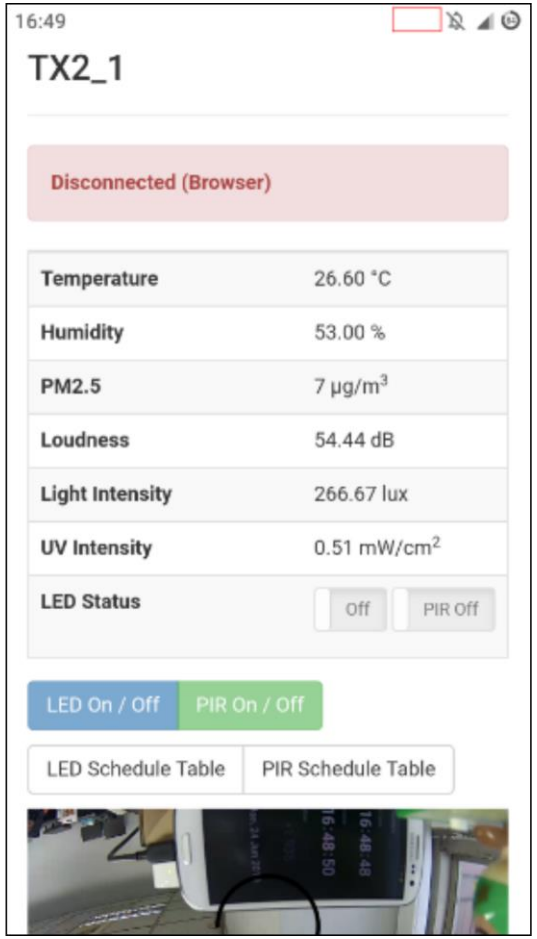
- Remove **USB-serial**
- Remove **RJ45**

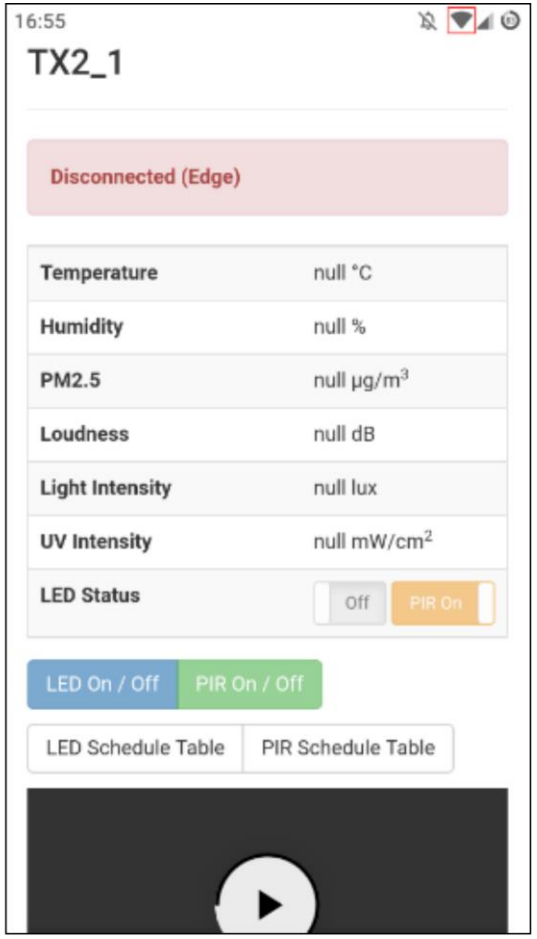


Edge Service Handling (Cloud Module)







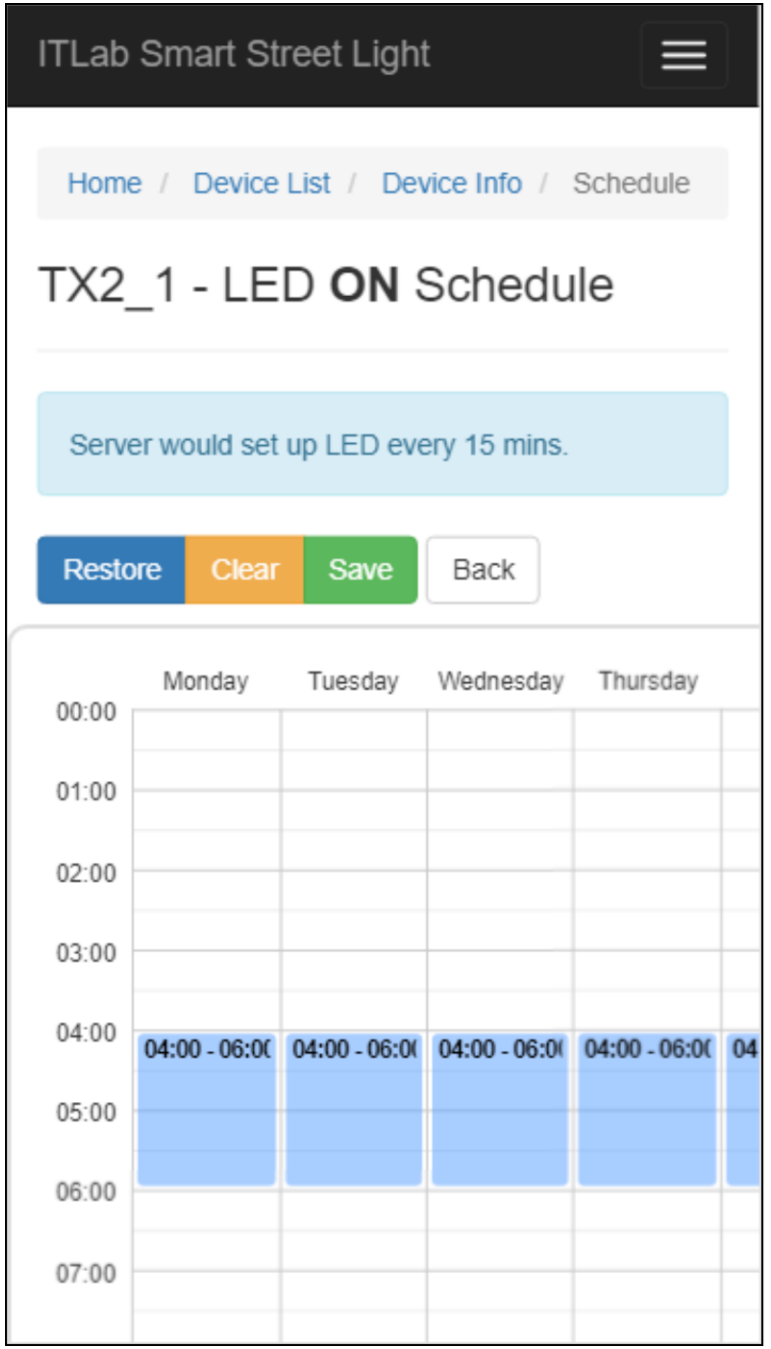


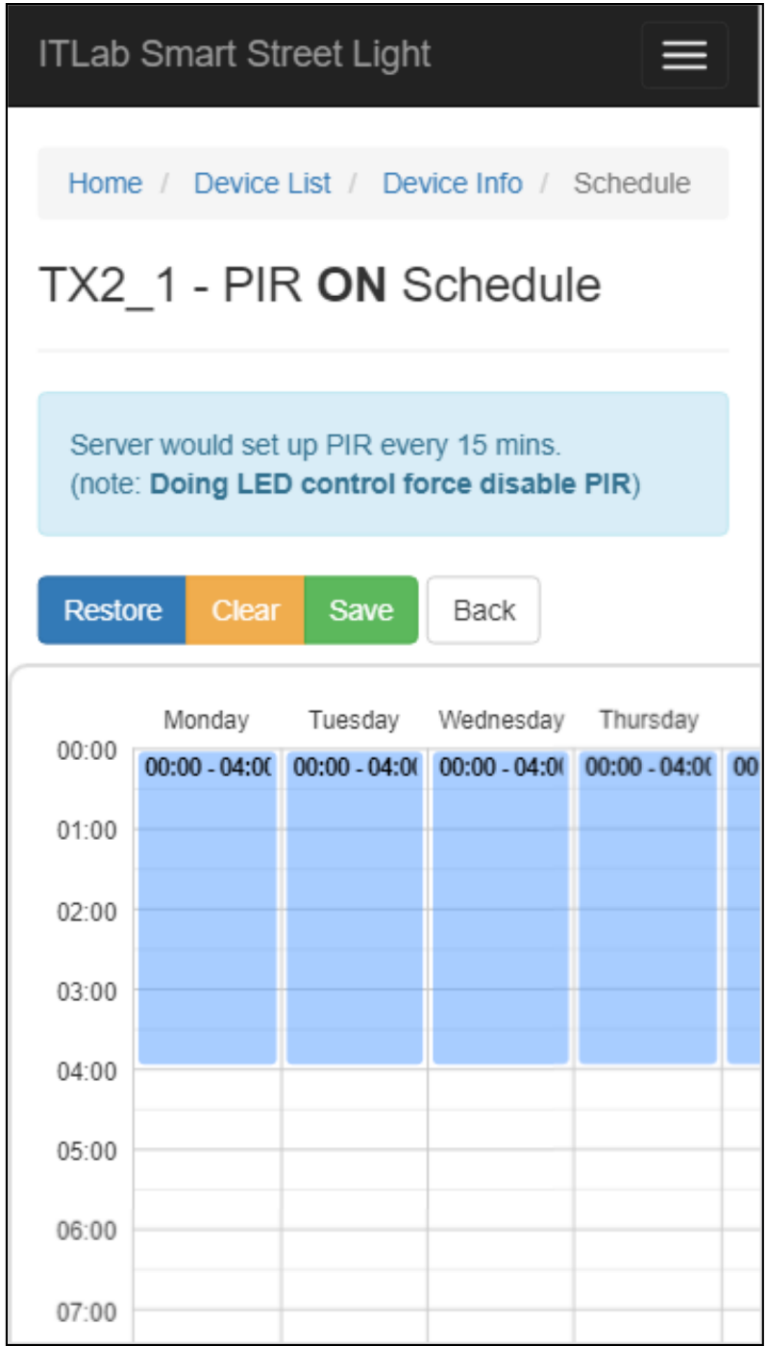
Edge Service Handling
(Cloud Module)

LED (PIR) **ON/OFF** Timetables

(Send CMDs to Edge)

- Every 15 min.
- LED > PIR



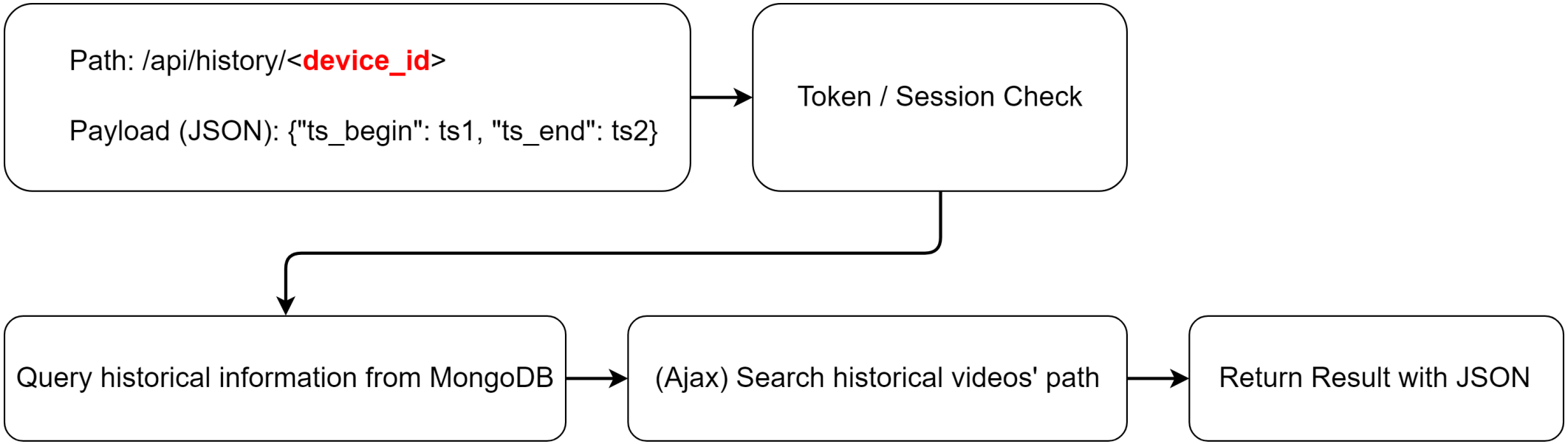


Historical Data Query – Cron Jobs

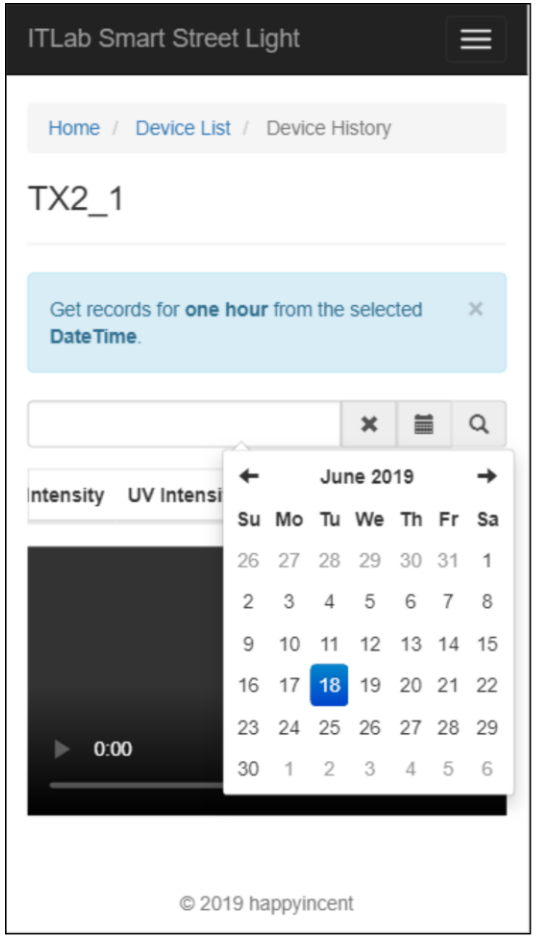
- Every 15 min.
  - Export current information from **Redis** into **MongoDB**

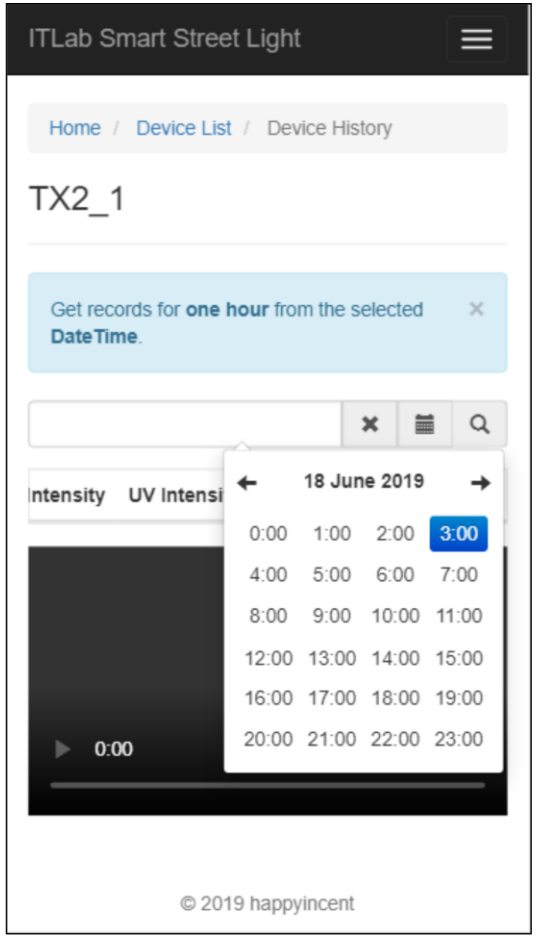
    (loop all devices and check timestamp to avoid duplication)
- Every day
  - Time sync through *ntpd*
  - Delete the oldest 20% historical videos if the disk usage is over 80%
  - Delete historical documents in MongoDB which are older than 1 year

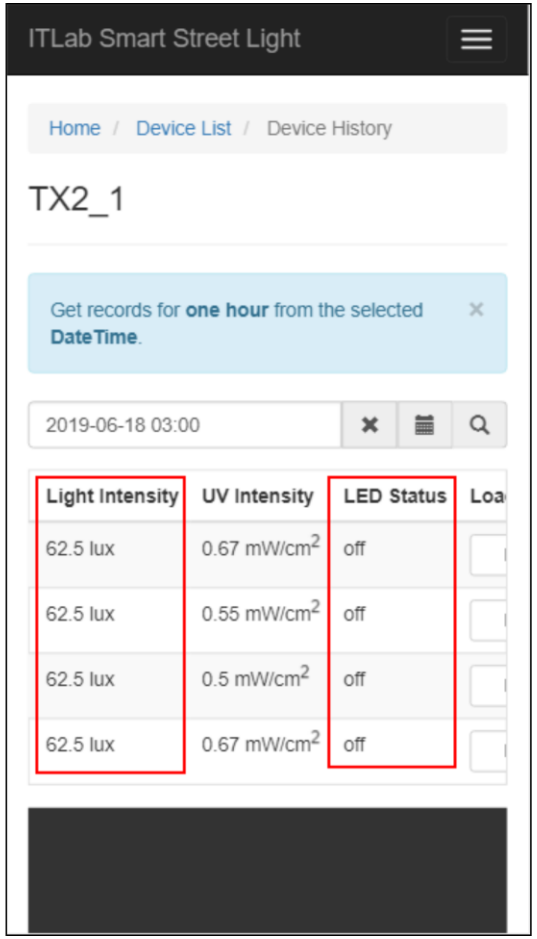
Historical Data Query **API**

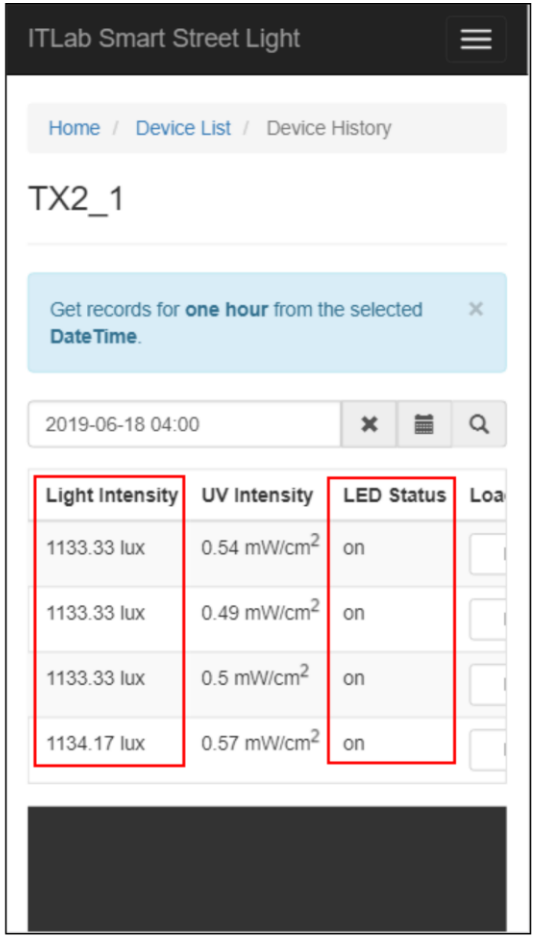


Historical Data Query **API** (Browser - Ajax)

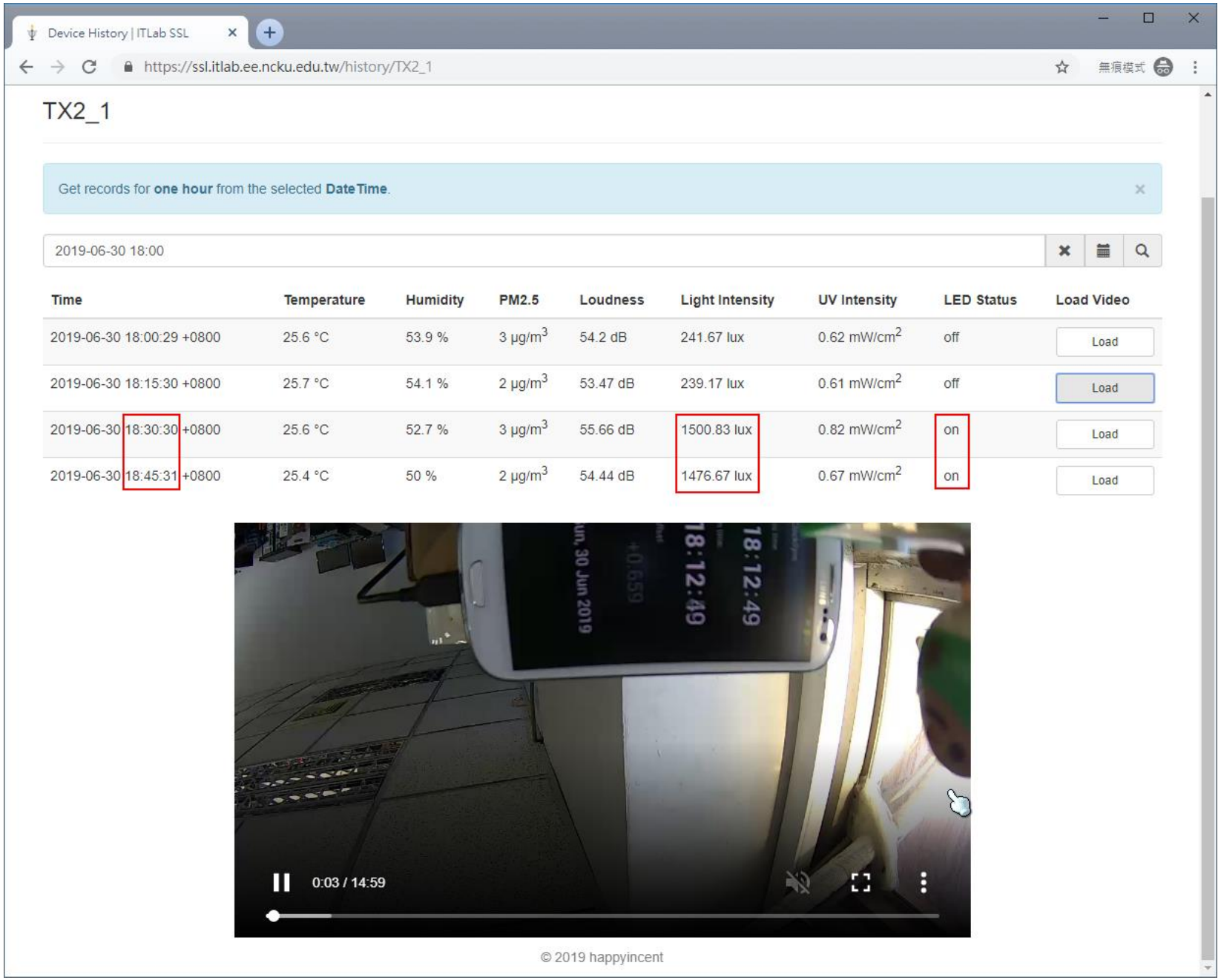








Historical Data Query **API** (Browser - Ajax)



Historical Data Query **API** (Open Data)

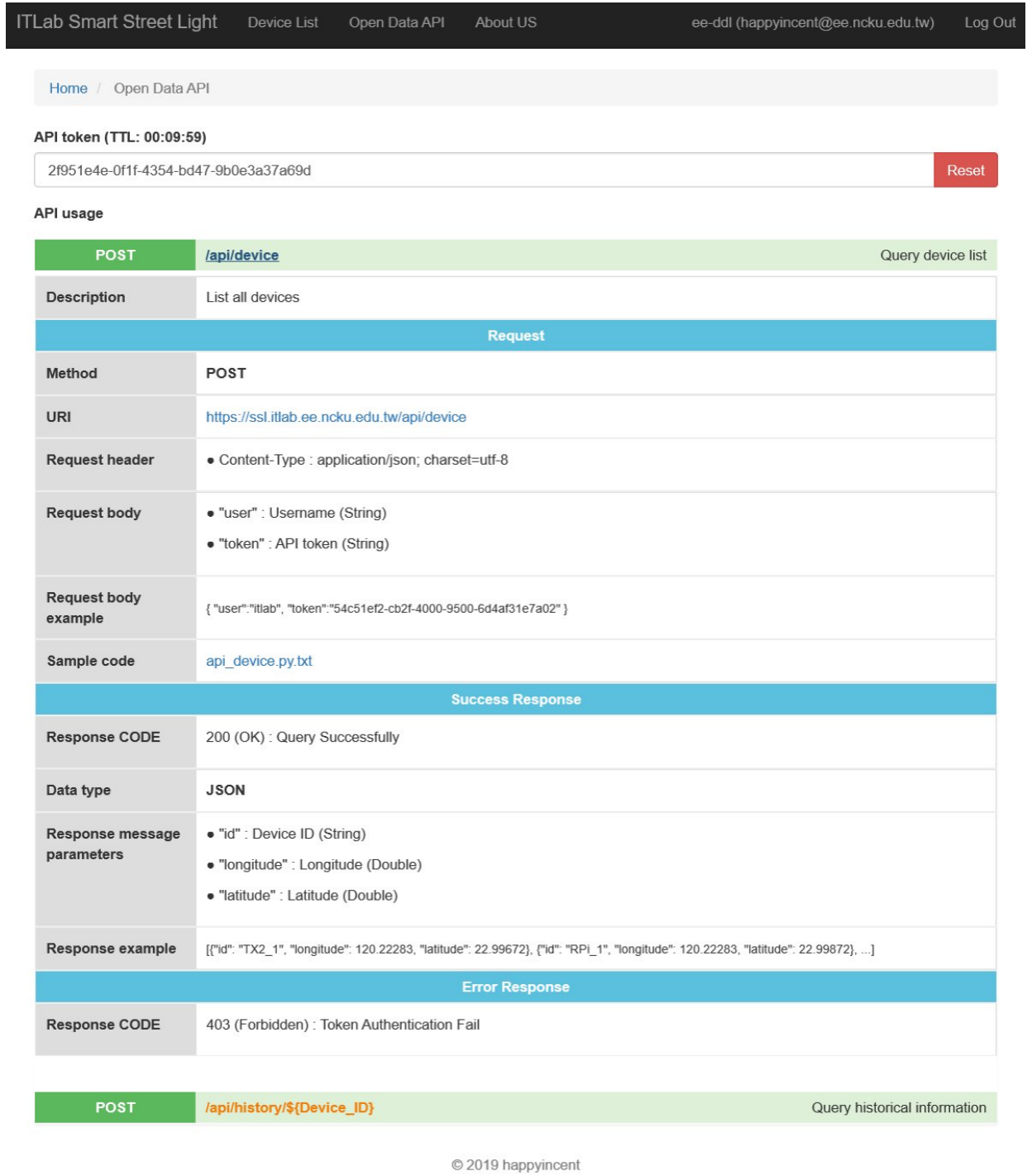
/api/device

{

“user”: username,

“token”: token

}



ITLab Smart Street Light | Device List | Open Data API | About US | ee-ddl (happyincent@ee.ncku.edu.tw) | Log Out

Home / Open Data API

API token (TTL: 00:09:59)

2f951e4e-0f1f-4354-bd47-9b0e3a37a69d Reset

API usage

POST	/api/device	Query device list
Description	List all devices	
Request		
Method	**POST**	
URI	https://ssl.itlab.ee.ncku.edu.tw/api/device	
Request header	• Content-Type : application/json; charset=utf-8	
Request body	• "user" : Username (String) • "token" : API token (String)	
Request body example	{ "user":"itlab", "token":"54c51ef2-cb2f-4000-9500-6d4af31e7a02" }	
Sample code	api_device.py.txt	
Success Response		
Response CODE	200 (OK) : Query Successfully	
Data type	**JSON**	
Response message parameters	• "id" : Device ID (String) • "longitude" : Longitude (Double) • "latitude" : Latitude (Double)	
Response example	[{"id": "TX2_1", "longitude": 120.22283, "latitude": 22.99672}, {"id": "RPi_1", "longitude": 120.22283, "latitude": 22.99872}, ...]	
Error Response		
Response CODE	403 (Forbidden) : Token Authentication Fail	

POST	/api/history/${Device_ID}	Query historical information

© 2019 happyincent

Historical Data Query **API** (Open Data)

/api/history/${Device_ID}

{

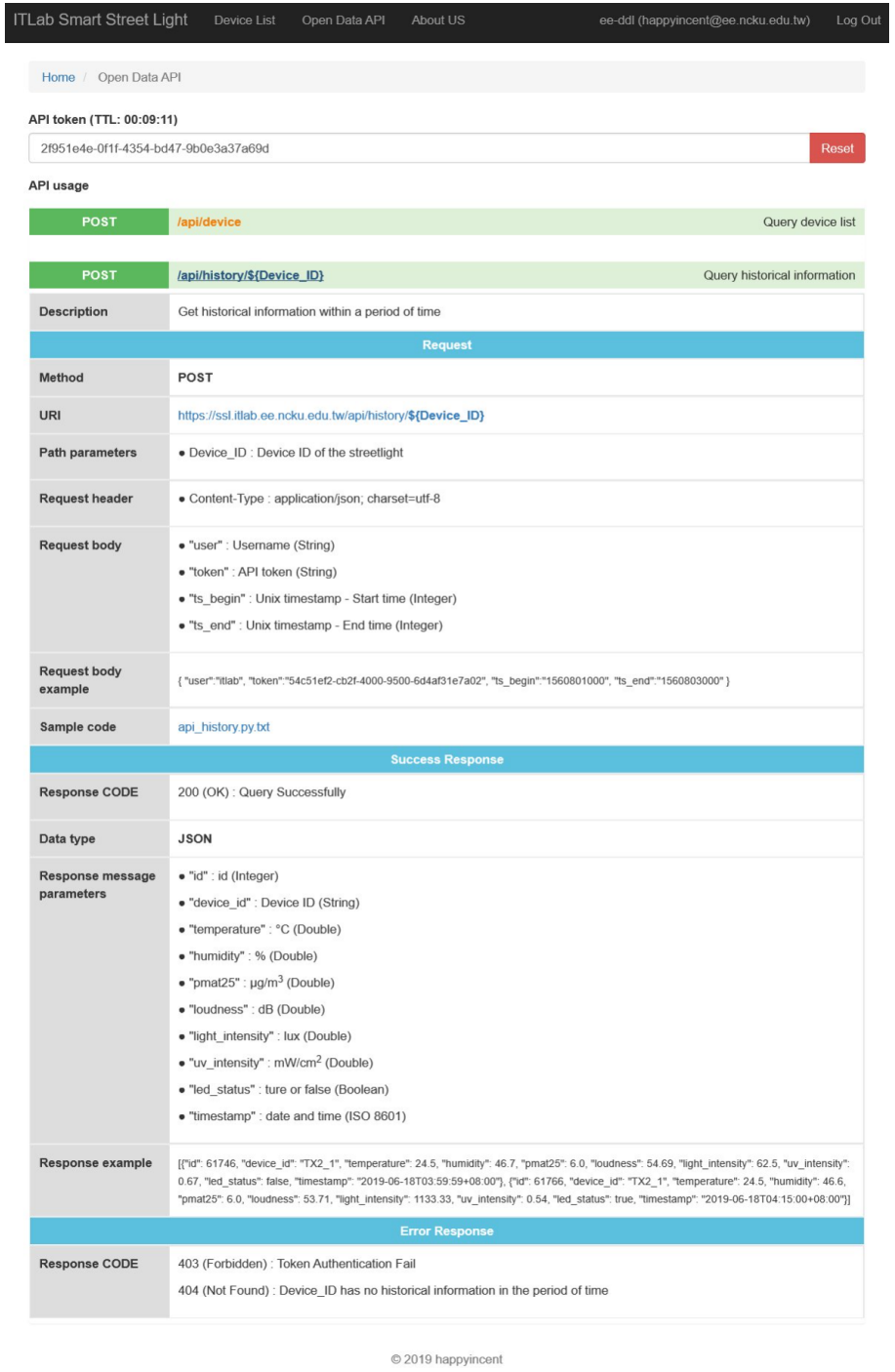
“user”: username,

“token”: token,

“ts_begin”: 1560801000,

“ts_end”: 1560803000,

}



ITLab Smart Street Light | Device List | Open Data API | About US | ee-ddl (happyincent@ee.ncku.edu.tw) | Log Out

Home / Open Data API

API token (TTL: 00:09:11)

2f951e4e-0f1f-4354-bd47-9b0e3a37a69d Reset

API usage

POST	/api/device	Query device list

POST	/api/history/${Device_ID}	Query historical information

Description	Get historical information within a period of time
Request	
Method	POST
URI	https://ssl.itlab.ee.ncku.edu.tw/api/history/**${Device_ID}**
Path parameters	• Device_ID : Device ID of the streetlight
Request header	• Content-Type : application/json; charset=utf-8
Request body	• "user" : Username (String) • "token" : API token (String) • "ts_begin" : Unix timestamp - Start time (Integer) • "ts_end" : Unix timestamp - End time (Integer)
Request body example	{ "user":"itlab", "token":"54c51ef2-cb2f-4000-9500-6d4af31e7a02", "ts_begin":"1560801000", "ts_end":"1560803000" }
Sample code	api_history.py.txt
Success Response	
Response CODE	200 (OK) : Query Successfully
Data type	JSON
Response message parameters	• "id" : id (Integer) • "device_id" : Device ID (String) • "temperature" : °C (Double) • "humidity" : % (Double) • "pmat25" : μg/m^3 (Double) • "loudness" : dB (Double) • "light_intensity" : lux (Double) • "uv_intensity" : mW/cm^2 (Double) • "led_status" : ture or false (Boolean) • "timestamp" : date and time (ISO 8601)
Response example	[{"id": 61746, "device_id": "TX2_1", "temperature": 24.5, "humidity": 46.7, "pmat25": 6.0, "loudness": 54.69, "light_intensity": 62.5, "uv_intensity": 0.67, "led_status": false, "timestamp": "2019-06-18T03:59:59+08:00"}, {"id": 61766, "device_id": "TX2_1", "temperature": 24.5, "humidity": 46.6, "pmat25": 6.0, "loudness": 53.71, "light_intensity": 1133.33, "uv_intensity": 0.54, "led_status": true, "timestamp": "2019-06-18T04:15:00+08:00"}]
Error Response	
Response CODE	403 (Forbidden) : Token Authentication Fail 404 (Not Found) : Device_ID has no historical information in the period of time

© 2019 happyincent

Outline

- Introduction
- Background and Related works
- System Design
  - Cloud Management System
  - Edge Services Orchestrator
- Experiment and Verification
- Conclusion and Future works

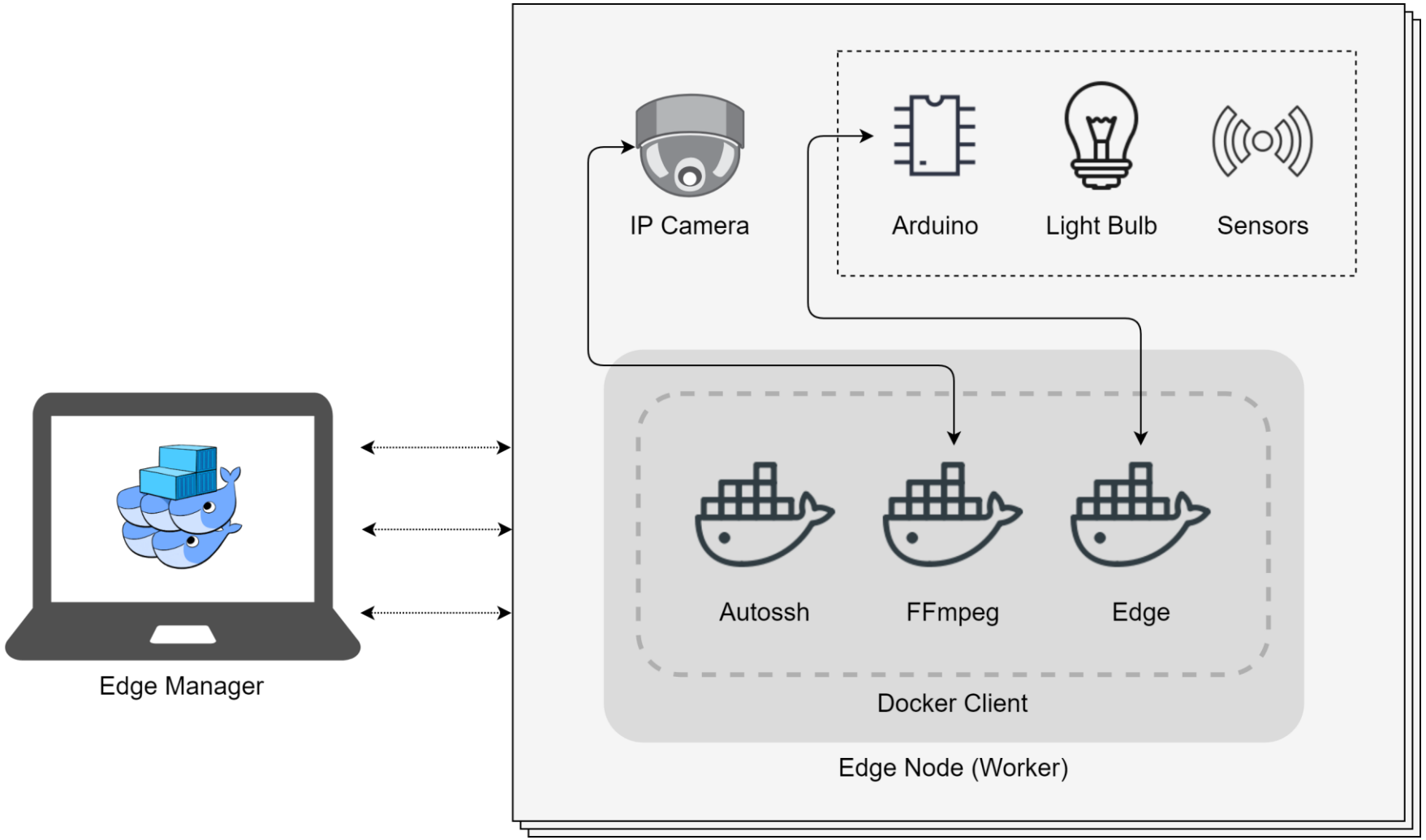
Edge Services Orchestrator

- Edge Nodes Architecture
  - Overview
  - Docker images, containers
- Edge Services
  - Secure Transmission Module
  - Video Streaming Module
  - Edge Service Module
- Edge Nodes Deployment
  - Single Node Deployment
  - Cluster Nodes Deployment

Edge Services Orchestrator

- Edge Nodes Architecture
  - Overview
  - Docker images, containers
- Edge Services
  - Secure Transmission Module
  - Video Streaming Module
  - Edge Service Module
- Edge Nodes Deployment
  - Single Node Deployment
  - Cluster Nodes Deployment

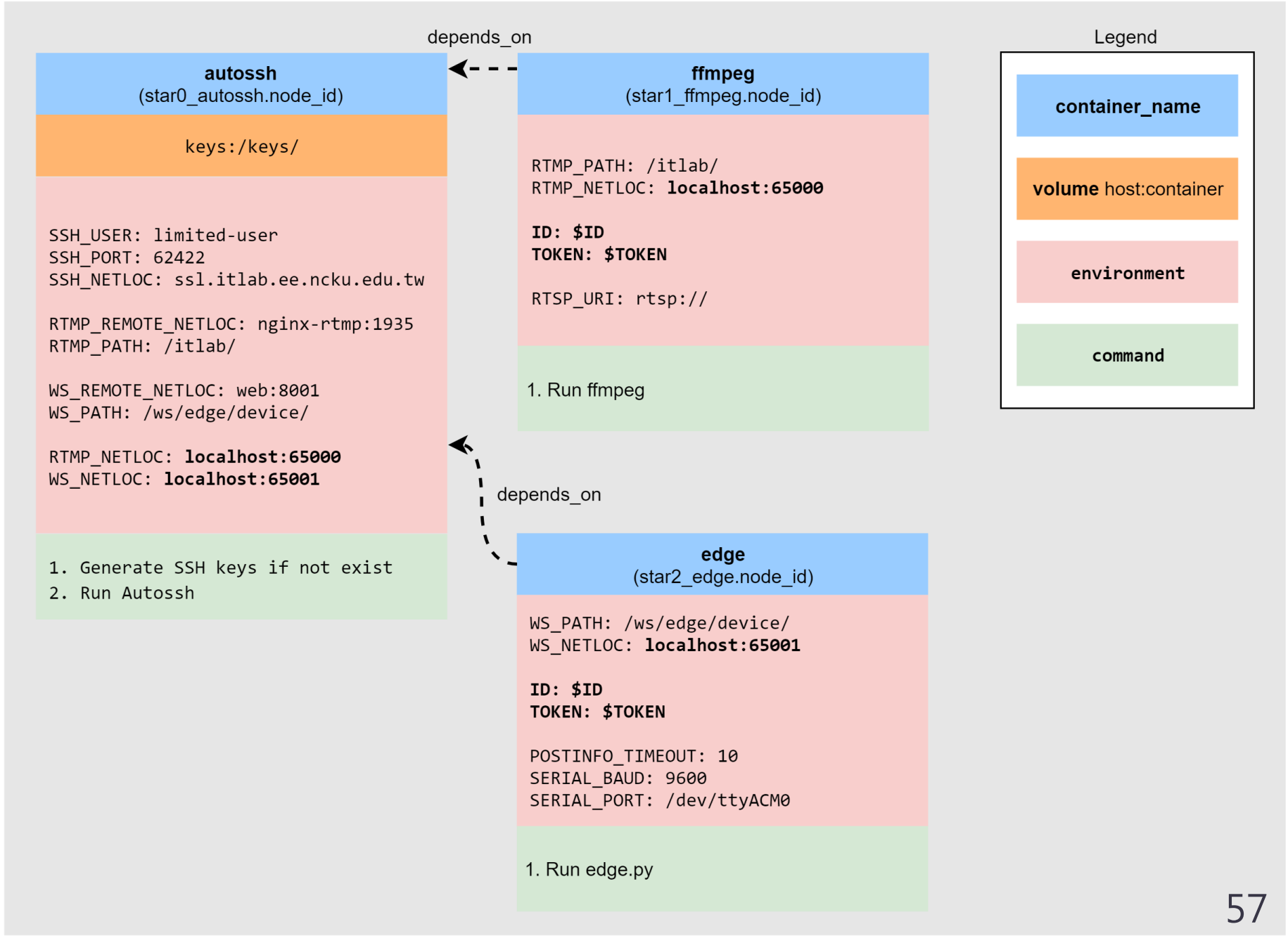
Edge Nodes - Overview



Edge Nodes - Docker Images

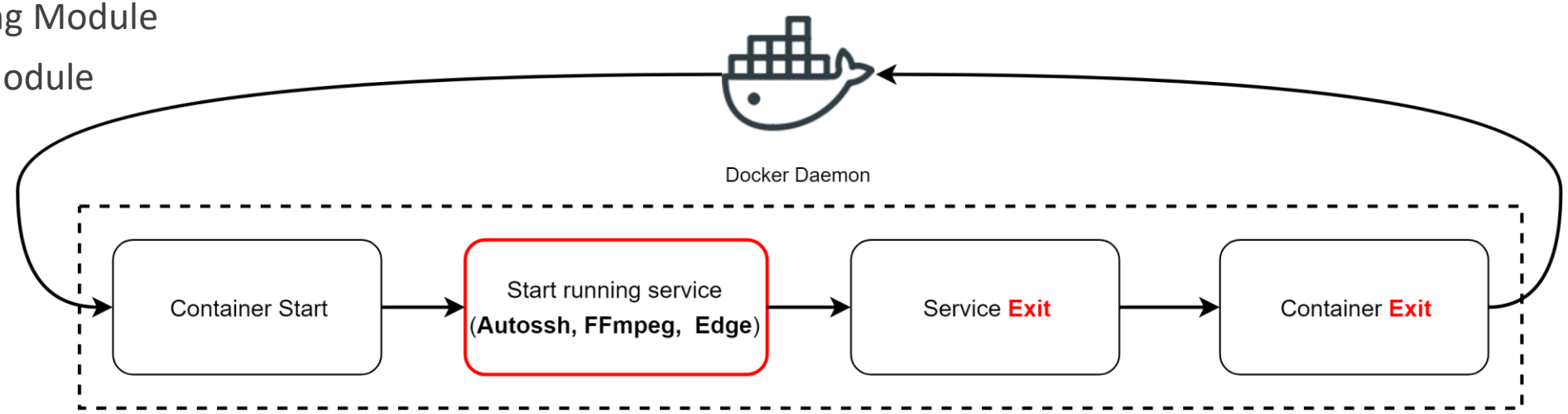
Image / Info	Base Image	Size (amd64)	Size (arm64)	Size (armv7)
Autossh	alpine:3.9	11.5MB	11.2MB	8.71MB
FFmpeg	alpine:3.9	57MB	47.9MB	35.6MB
Edge	python:3.7.3-alpine3.9	93.7MB	100MB	78MB

Edge Nodes - Docker Containers

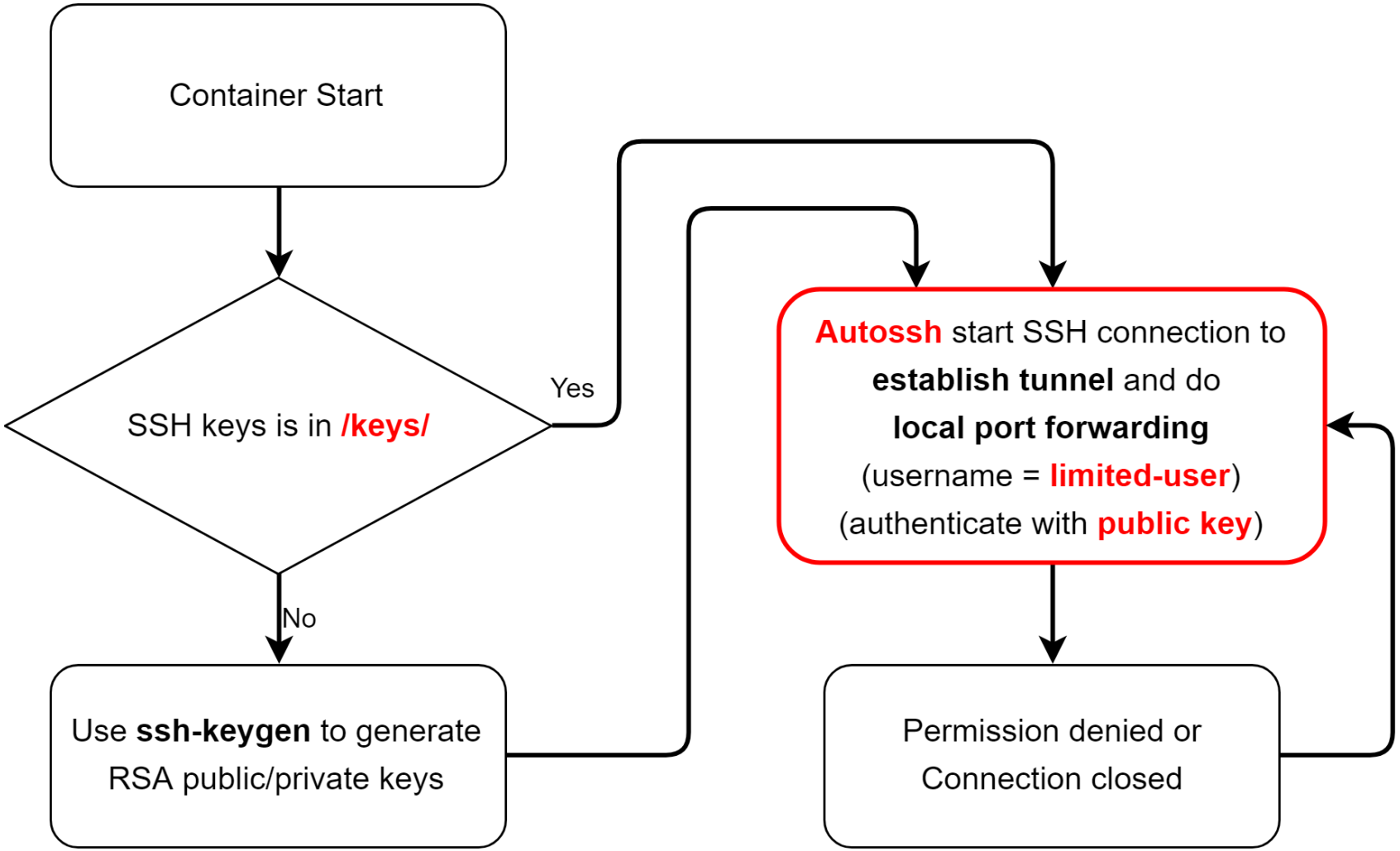


Edge Services Orchestrator

- Edge Nodes Architecture
  - Overview
  - Docker images, containers
- Edge Services
  - Secure Transmission Module
  - Video Streaming Module
  - Edge Service Module
- Edge Nodes Deployment
  - Single Node Deployment
  - Cluster Nodes Deployment



Secure Transmission Module



Video Streaming Module

- RTSP → FFmpeg (with token) → (SSH Tunnel) → Nginx-RTMP

```
ffmpeg \
-y -loglevel warning \
-use_wallclock_as_timestamps 1 \
-flags global_header \
-fflags +genpts \
-r 15 -rtsp_transport tcp \
-i $$RTSP_URI \
-vcodec copy -an \
-bsf:v h264_mp4toannexb \
-tune zerolatency \
-f flv -rtmp_live live -rtmp_buffer 0 -timeout 0 \
rtmp://${RTMP_NETLOC}${RTMP_PATH}$$ID?token=$$TOKEN
```

Edge Service Module – **Edge Container**

- USB-serial → Edge → WebSocket (with token) → (SSH Tunnel) → Web

(Threads)

- **SerialHandler** (Arduino to Edge)
  - Forward INFO to Cloud
- **PostInfoHandler** (Edge to Arduino)
  - Ask Arduino for the latest INFO every 10 s
- **MessageHandler** (Cloud to Edge)
  - Forward CMDs to Arduino

Edge Service Module – **Arduino**

(LOOP)

- Read Serial CMDs (Edge to Arduino)
  - Turn On/Off LED
  - Turn On/Off PIR
  - Change PIR timeout
- PIR Control (if enable)
  - Sensed → Turn On LED → Turn Off LED (after timeout)
- Query Sensors' value (every 1 s)

Edge Services Orchestrator

- Edge Nodes Architecture
  - Overview
  - Docker images, containers
- Edge Services
  - Secure Transmission Module
  - Video Streaming Module
  - Edge Service Module
- Edge Nodes Deployment
  - Single Node Deployment
  - Cluster Nodes Deployment

Single Node Deployment

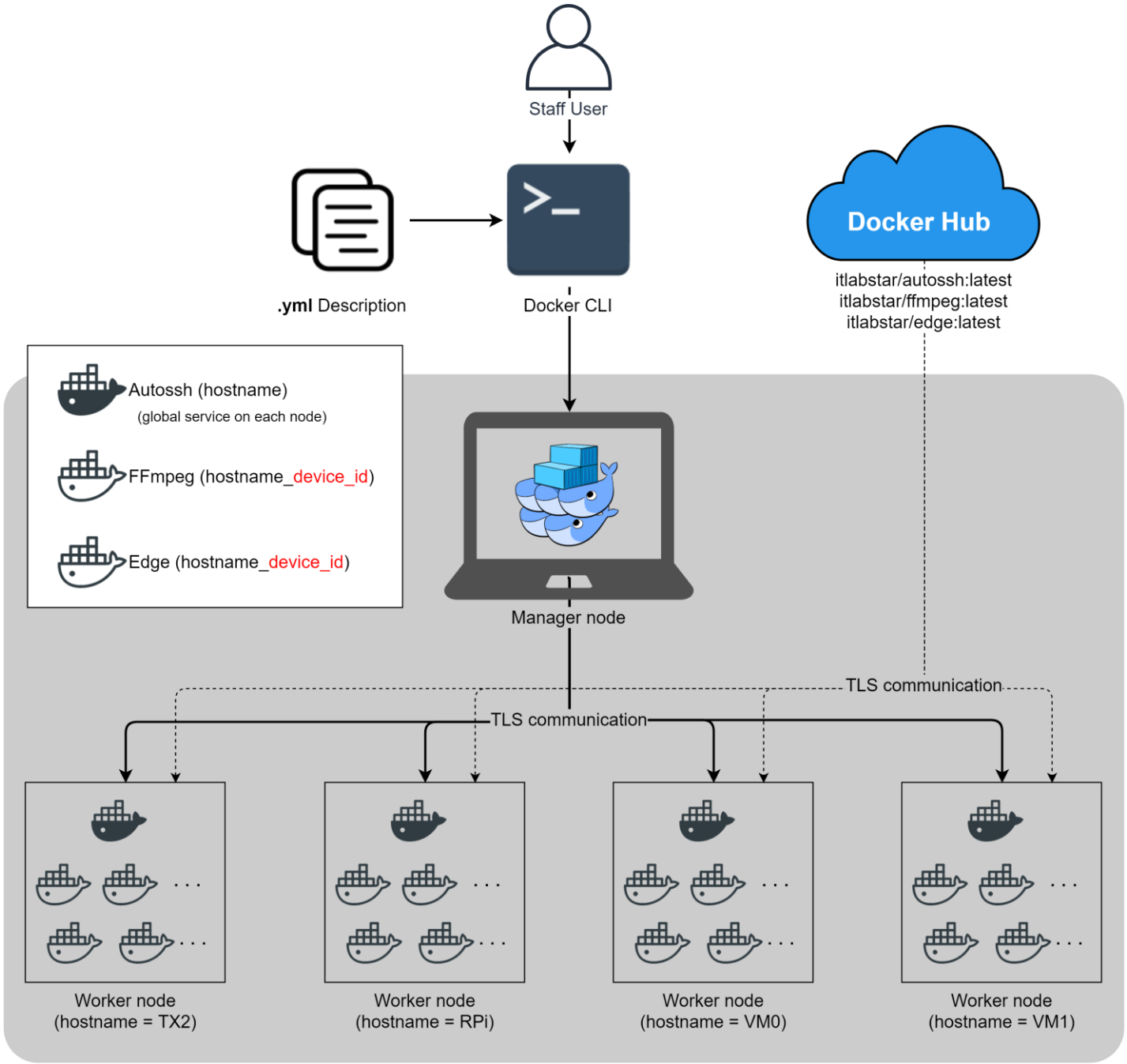
- Build images locally
  - Dockerfiles
- Run containers
  - Compose files (.ymls), Environment files (.env, star-single.env)

```
nvidia@TX2:~/Desktop/star-edge$ echo "$(cat .env)"
#### Edge Service Parameters ####
SSH_USER=limited-user
SSH_PORT=62422
SSH_NETLOC=ssl.itlab.ee.ncku.edu.tw

RTMP_REMOTE_NETLOC=nginx-rtmp:1935
RTMP_PATH=/itlab/

WS_REMOTE_NETLOC=web:8001
WS_PATH=/ws/edge/device/

RTMP_NETLOC=localhost:65000
WS_NETLOC=localhost:65001
```

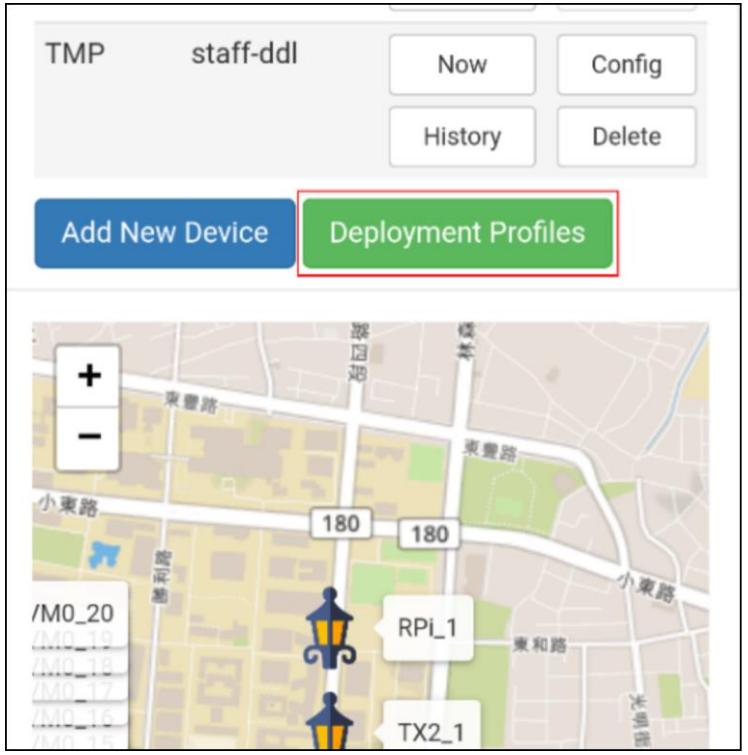
```
nvidia@TX2:~/Desktop/star-edge$ echo "$(cat star-single.env)"
#### Device Parameters ####
ID=TX2_1
TOKEN=ef8a6d2d-cd2c-449f-a455-5d646861b863

POSTINFO_TIMEOUT=10
SERIAL_BAUD=9600
SERIAL_PORT=/dev/ttyACM0

RTSP_URI=rtsp://[redacted]@10.27.164.152/live3.sdp
```

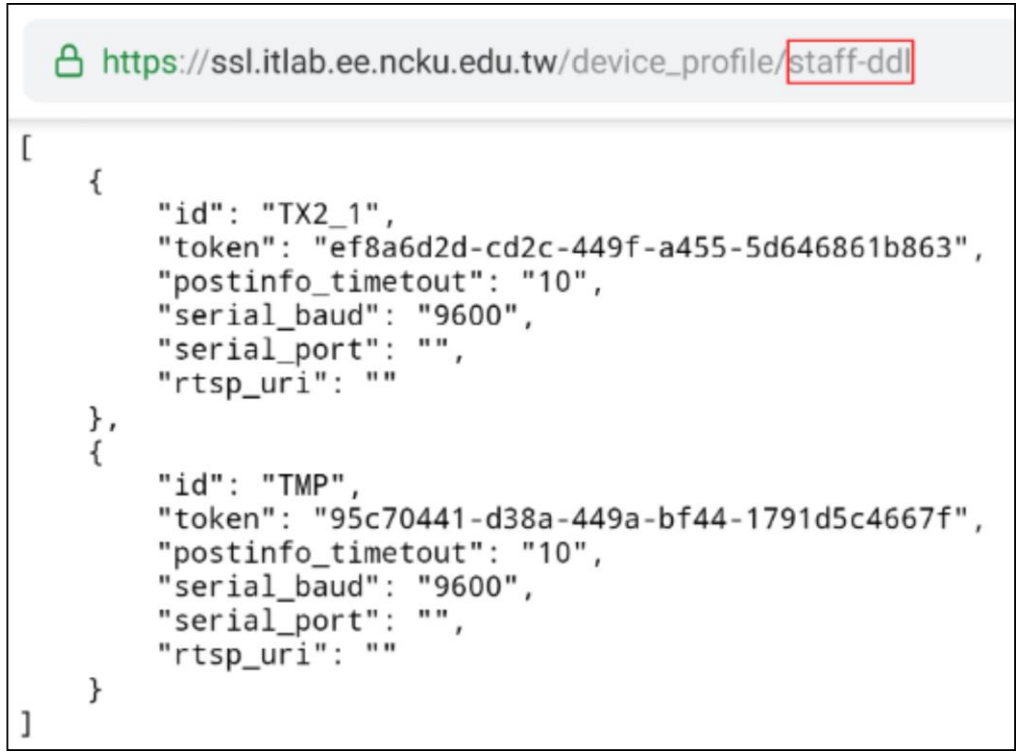
Cluster Nodes Deployment

- Docker Swarm
- Docker Compose (.ymls)
- Docker buildx (**cross-platform** images)
- Docker Hub



Compose File Generator

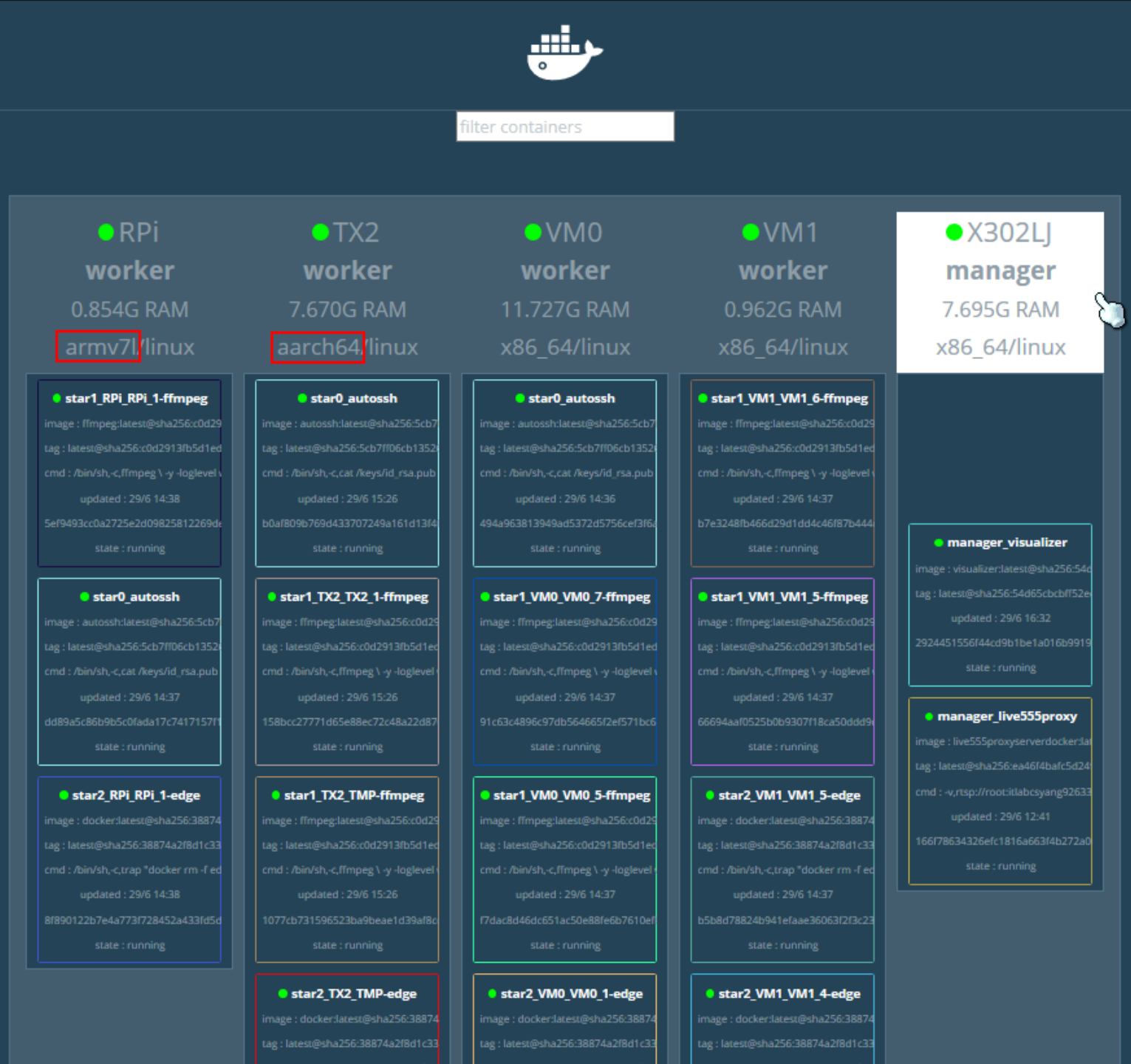




Compose File Generator

```
itlab@X302LJ:~/Desktop/star-edge$ rm node/yml/*
itlab@X302LJ:~/Desktop/star-edge$ ls node/profile/
RPi.json  TX2.json  VM0.json  VM1.json
itlab@X302LJ:~/Desktop/star-edge$ echo "$(cat node/profile/TX2.json)"
{
    "NODE_NAME": "TX2",
    "DEVICES": [
        {
            "id": "TX2_1",
            "token": "ef8a6d2d-cd2c-449f-a455-5d646861b863",
            "postinfo_timetout": "10",
            "serial_baud": "9600",
            "serial_port": "/dev/ttyACM0",
            "rtsp_uri": "rtsp://[redacted]@10.27.164.152/live3.sdp"
        },
        {
            "id": "TMP",
            "token": "95c70441-d38a-449a-bf44-1791d5c4667f",
            "postinfo_timetout": "10",
            "serial_baud": "9600",
            "serial_port": "",
            "rtsp_uri": "rtsp://[redacted]/proxyStream"
        }
    ]
}
itlab@X302LJ:~/Desktop/star-edge$ ./node/generate.sh `ls $PWD/node/profile/*.json`
Gernerating YAMLs from the profile: /home/itlab/Desktop/star-edge/node/profile/RPi.json ...
Gernerating YAMLs from the profile: /home/itlab/Desktop/star-edge/node/profile/TX2.json ...
Gernerating YAMLs from the profile: /home/itlab/Desktop/star-edge/node/profile/VM0.json ...
Gernerating YAMLs from the profile: /home/itlab/Desktop/star-edge/node/profile/VM1.json ...
itlab@X302LJ:~/Desktop/star-edge$ tree -P "TX2*" node/yml/
node/yml/
├── TX2_TMP-edge.yml
├── TX2_TMP-ffmpeg.yml
├── TX2_TX2_1-edge.yml
└── TX2_TX2_1-ffmpeg.yml
```

Swarm Visualizer



Outline

- Introduction
- Background and Related works
- System Design
- Experiment and Verification
- Conclusion and Future works

Environment

(Cloud Server)	HP Z600
CPU	16 x 2.27 GHz
RAM	24 GB
OS	Xubuntu 16.04.6
Software	Docker 18.09.7, Docker Compose 1.24.0

(Other Devices)	Description
Arduino Mega 2560	Microcontroller
VIVOTEK FD8169A	IP Camera
ASUS Laptop X302LJ	• Swarm Manager • RTSP Proxy Server

(Edge Nodes)	Nvidia Jetson TX2	Raspberry Pi 3B	VM0 (ESXi 6.0)	VM1 (ESXi 5.5)
CPU	2 x2 + 4x 2 GHz	4 x 1.2 GHz	8 x 3.7 GHz	8 x 3.4 GHz
RAM	8 GB	1 GB	12 GB	2 GB
OS	Ubuntu 16.04.6	Raspbian 9.9	Xubuntu 16.04.6	Ubuntu server 18.04.2
Software	Docker 18.09.7	Docker 18.09.0	Docker 18.09.6	Docker 18.09.6

Backup & Restore Cloud Server

- Start time ≒ 21 s
- Stop time ≒ 35 s
- Size (with DB & Code) ≒ 500 MB

```
itlab@Z600:~/Desktop/star-eval$ ./backup.sh
Stop all Docker containers ...

real    0m34.724s
user    0m1.165s
sys     0m0.119s

OK

Backup star project ...

real    0m15.012s
user    0m14.671s
sys     0m1.510s

OK

Backup Docker images ...

real    0m29.452s
user    0m53.618s
sys     0m2.806s

OK

List tar files ...
213M    star.tar.gz
74M     certbot:star.tar.gz
50M     mongo:star.tar.gz
38M     nginx-rtmp:star.tar.gz
7.3M    nginx:star.tar.gz
17M     redis:star.tar.gz
18M     sshd:star.tar.gz
83M     web:star.tar.gz
itlab@Z600:~/Desktop/star-eval$
```

```
itlab@Z600:~/Desktop/star-eval$ ./restore.sh
Remove star project ...
OK

Remove Docker images ...
OK

Restore star project ...

real    0m4.318s
user    0m4.302s
sys     0m1.842s

OK

Restore Docker images ...

real    0m11.014s
user    0m10.863s
sys     0m1.755s

OK

Start all Docker containers ...

real    0m20.819s
user    0m1.326s
sys     0m0.099s

OK
itlab@Z600:~/Desktop/star-eval$
```

Secure Transmission Verification

Dataflow	Transmission	Encryption	Authentication
Cloud → Browser (all requests) (hls key, vod mp4)	TCP / HTTP	HTTPS	Session
Cloud → Browser (edge service)	TCP / WebSocket	HTTPS	Session
Edge → Cloud (edge service)	TCP / WebSocket	SSH tunnel	Token
Edge → Cloud (streaming)	TCP / RTMP	SSH tunnel	Token
IP Camera → Edge (streaming)	TCP / RTSP	None **(LAN)**	Fixed password

Netfilter Logs

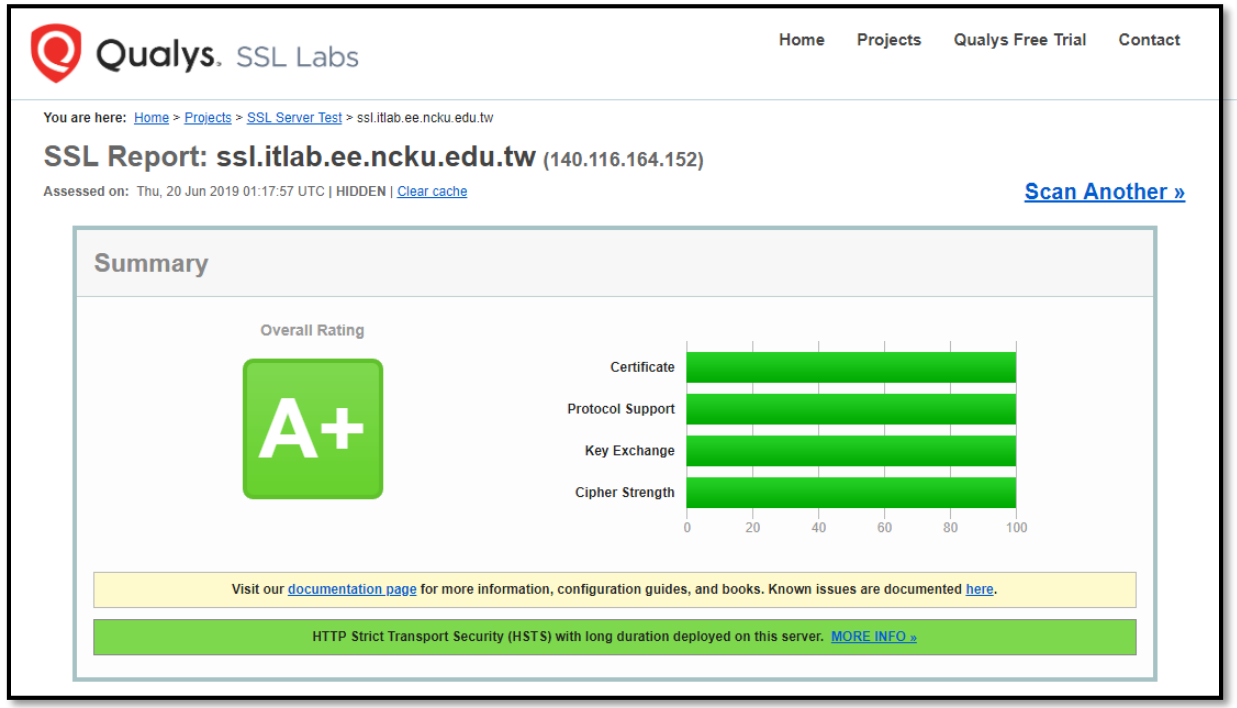
- WAN to Container (Internet → SSHD)

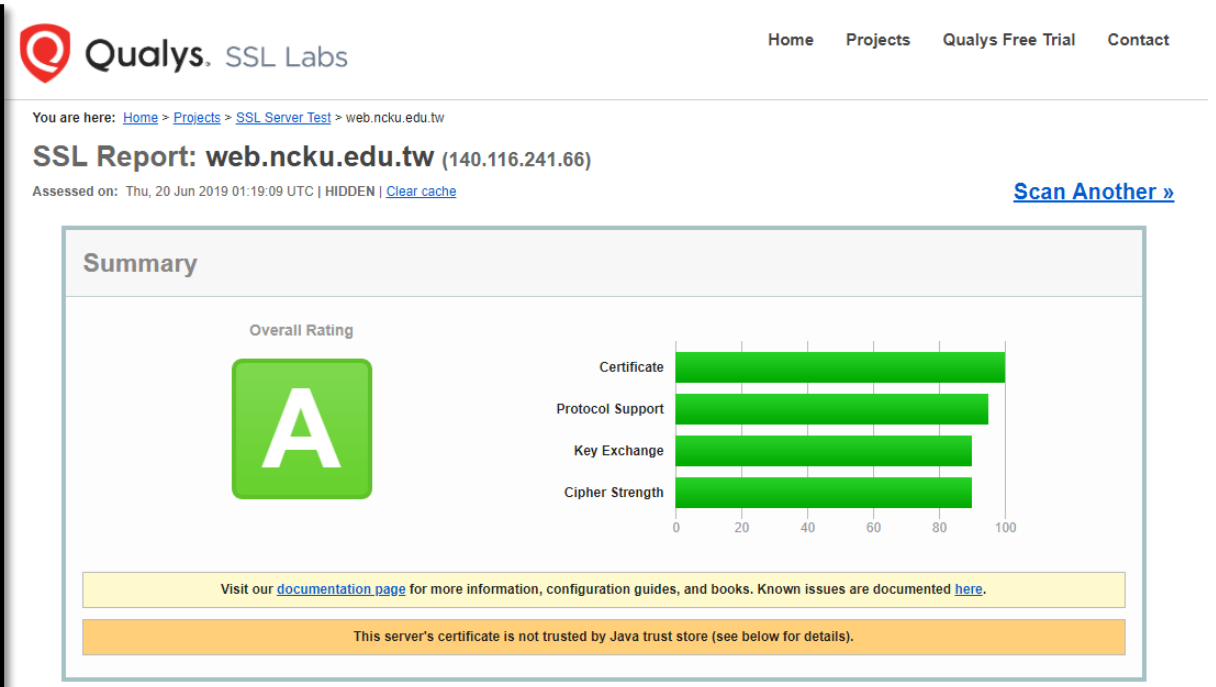
```
itlab@Z600:~$ cat log | sed -e 's|ctc/iptable/||g' -e 's|wtc/iptable/||g' -e 's/PHYSIN=[^ ]*//g' -e 's/PHYSOUT=[^ ]*//g' -e
^ ]*//g' -e 's/ID=[^ ]*//g' -e 's/DF[^ ]*//g' -e 's/ \+/,/g' | column -t -s,
mangle-PREROUTE   IN=enp1s0  OUT=                 SRC=10.27.164.153  DST=140.116.164.152  PROTO=TCP  SPT=59652  DPT=62422
nat-PREROUTE      IN=enp1s0  OUT=                 SRC=10.27.164.153  DST=140.116.164.152  PROTO=TCP  SPT=59652  DPT=62422
mangle-FORWARD    IN=enp1s0  OUT=br-73672de335db  SRC=10.27.164.153  DST=172.18.0.5       PROTO=TCP  SPT=59652  DPT=22
filter-forward    IN=enp1s0  OUT=br-73672de335db  SRC=10.27.164.153  DST=172.18.0.5       PROTO=TCP  SPT=59652  DPT=22
mangle-POSTROUTE  IN=        OUT=br-73672de335db  SRC=10.27.164.153  DST=172.18.0.5       PROTO=TCP  SPT=59652  DPT=22
nat-POSTROUTE     IN=        OUT=br-73672de335db  SRC=10.27.164.153  DST=172.18.0.5       PROTO=TCP  SPT=59652  DPT=22
```

- Container to Container (SSHD → Web, Web → Redis)

```
itlab@Z600:~$ cat log | sed -e 's|ctc/iptable/||g' -e 's|wtc/iptable/||g' -e 's/PHYSIN=[^ ]*//g' -e 's/PHYSOUT=[^ ]*//g' -e
^ ]*//g' -e 's/ID=[^ ]*//g' -e 's/DF[^ ]*//g' -e 's/ \+/,/g' | column -t -s,
mangle-PREROUTE   IN=br-73672de335db  OUT=                 SRC=172.18.0.5  DST=172.18.0.7  PROTO=TCP  SPT=35806  DPT=8001
filter-forward    IN=br-73672de335db  OUT=br-73672de335db  SRC=172.18.0.5  DST=172.18.0.7  PROTO=TCP  SPT=35806  DPT=8001
mangle-POSTROUTE  IN=                 OUT=br-73672de335db  SRC=172.18.0.5  DST=172.18.0.7  PROTO=TCP  SPT=35806  DPT=8001
mangle-PREROUTE   IN=br-73672de335db  OUT=                 SRC=172.18.0.7  DST=172.18.0.6  PROTO=TCP  SPT=45866  DPT=6379
filter-forward    IN=br-73672de335db  OUT=br-73672de335db  SRC=172.18.0.7  DST=172.18.0.6  PROTO=TCP  SPT=45866  DPT=6379
mangle-POSTROUTE  IN=                 OUT=br-73672de335db  SRC=172.18.0.7  DST=172.18.0.6  PROTO=TCP  SPT=45866  DPT=6379
```

SSL Report

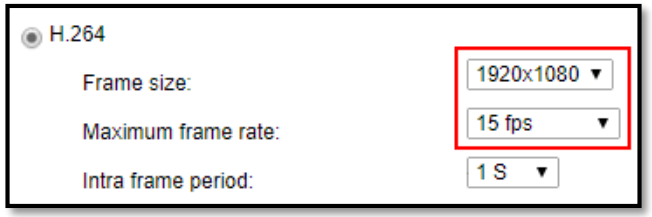




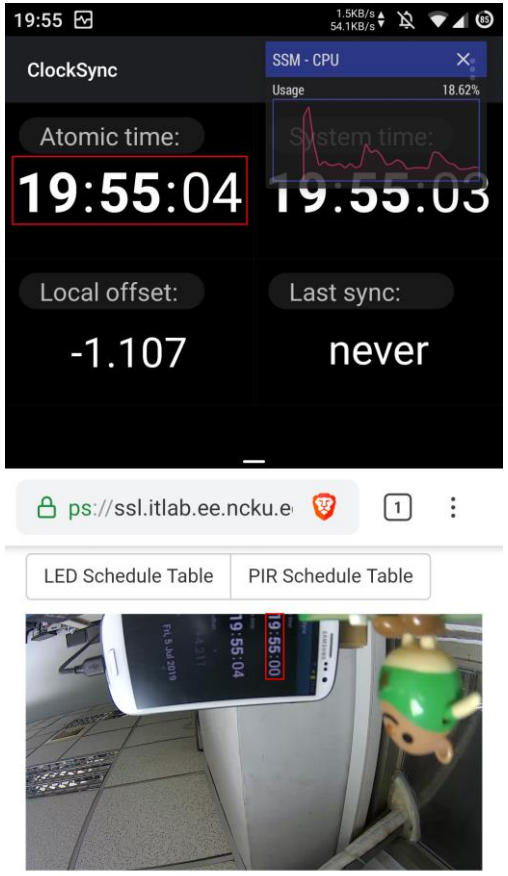
https://www.ssllabs.com/ssltest/

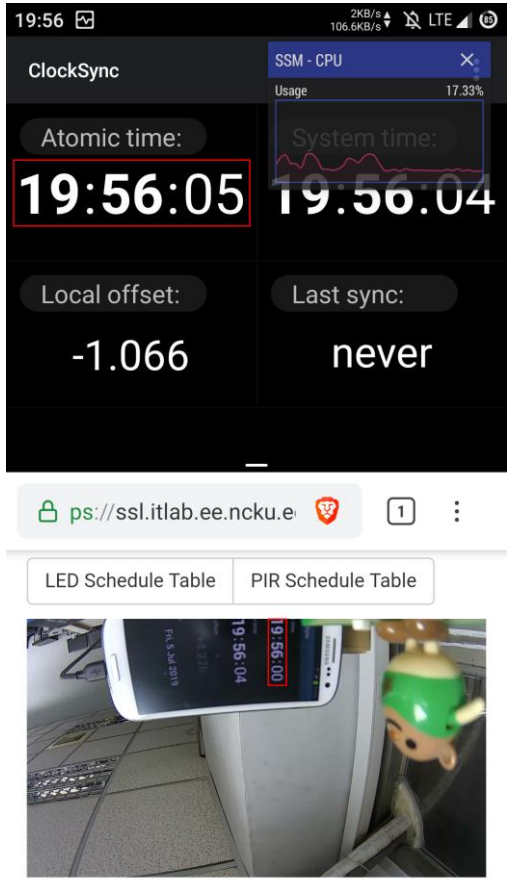
Video Latency in Live Streaming

- (POE Switch in **LAN**)
- (Android 9.0)
- CPU ≒ 18 %
- Wi-Fi ≒ 4 s
- LTE ≒ 5s

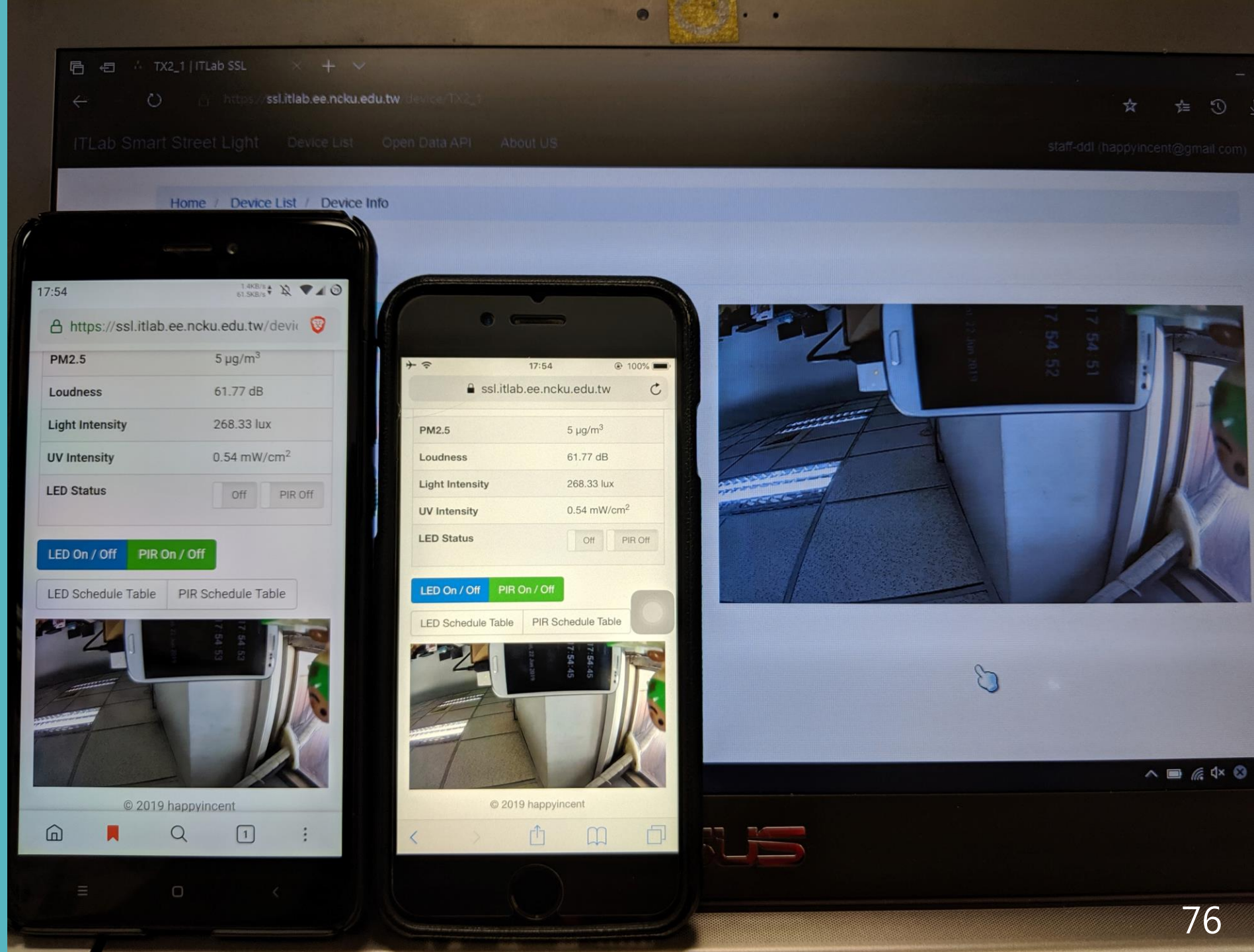



```
hls on;
hls_path /tmp/hls;

hls_playlist_length 3s;
hls_fragment 1s;

hls_keys on;
hls_key_path /tmp/key;
hls_key_url https://ssl.itlab.ee.ncku.edu.tw/key/;
hls_fragments_per_key 50;
```

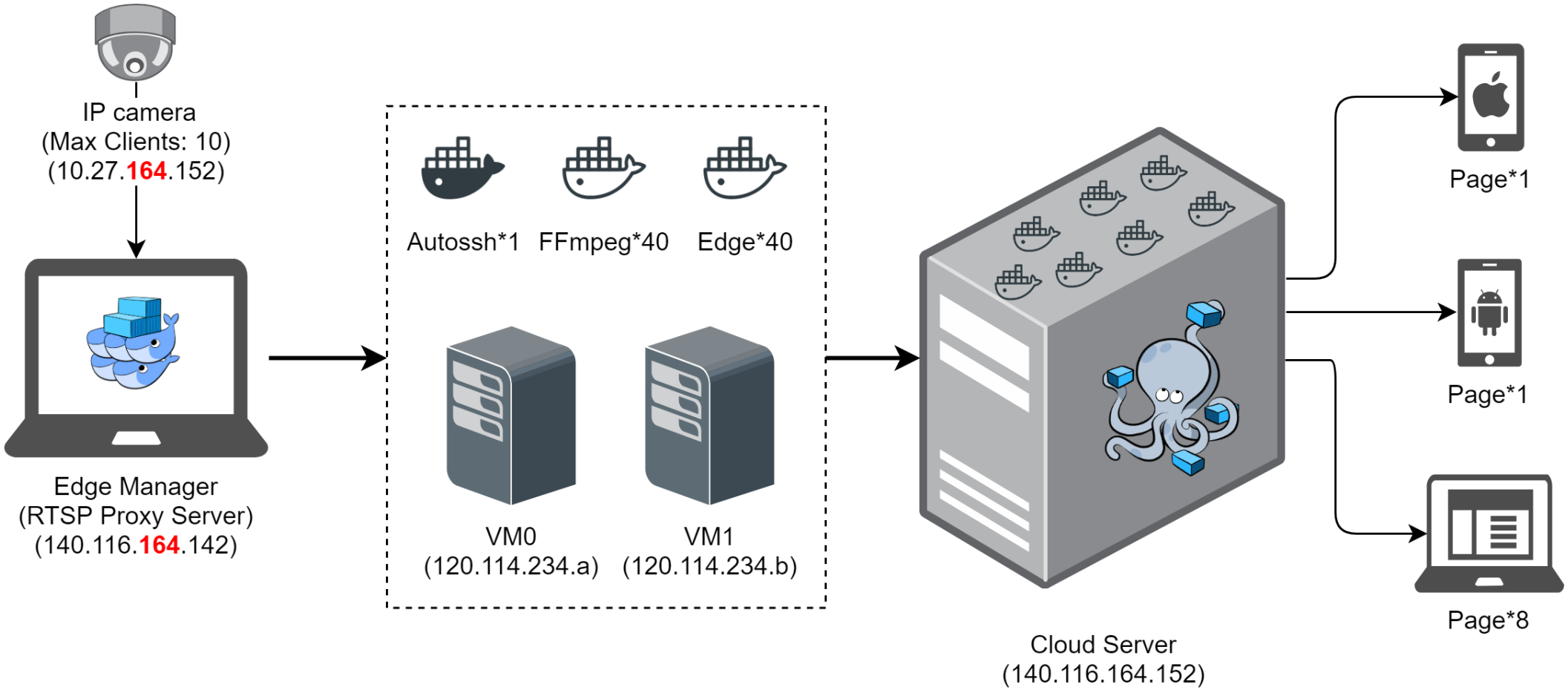




Cross-Platform Compatibility



Performance of Cloud Server



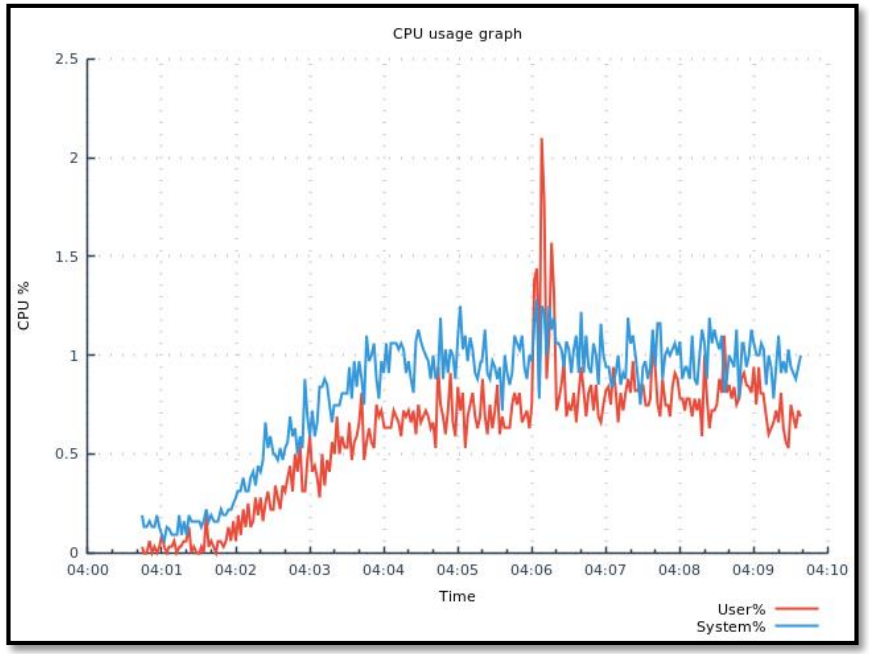
Performance of Cloud Server

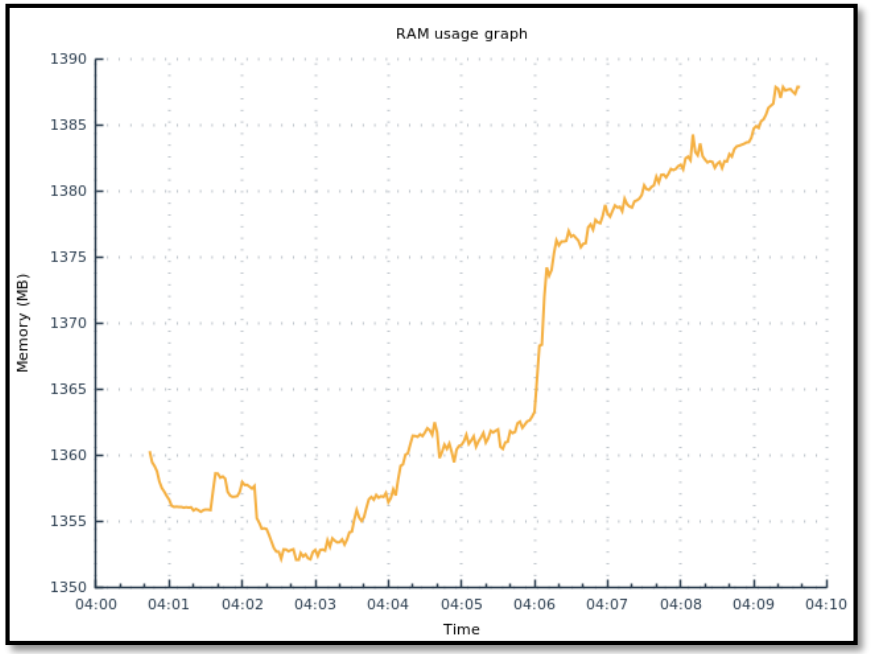
04:02 ~ :

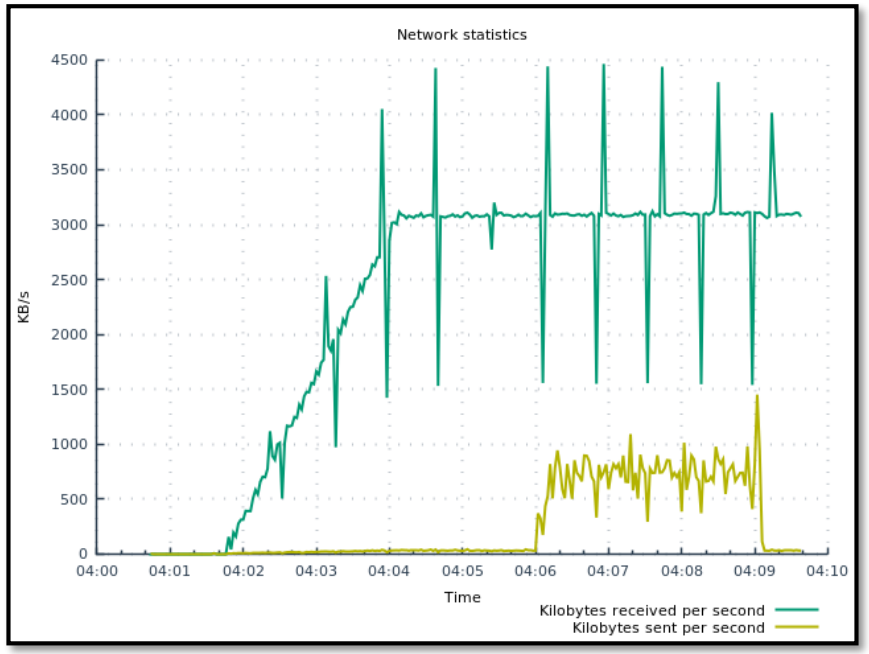
40 edge nodes

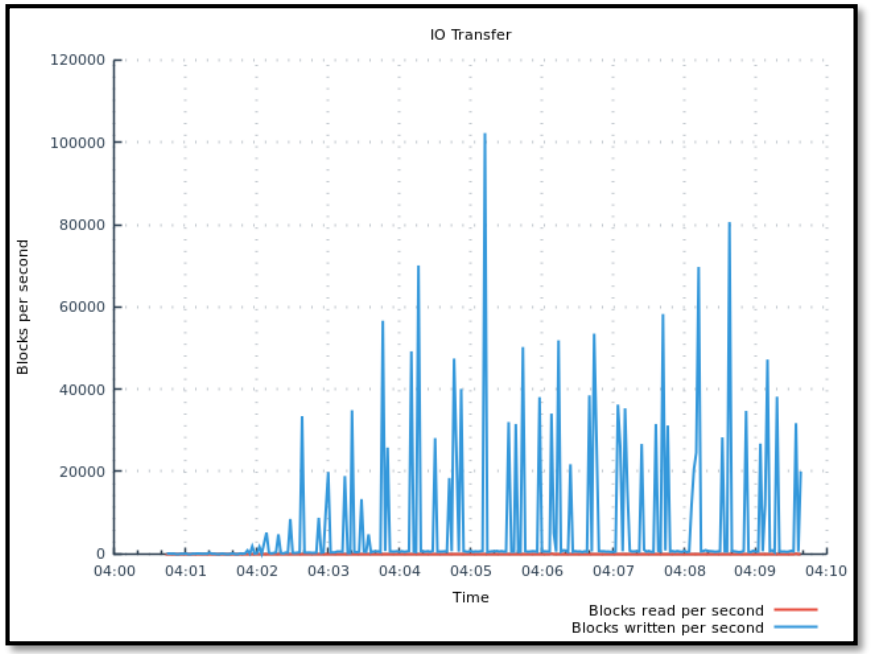
04:06 ~ 04:09:

10 browser clients







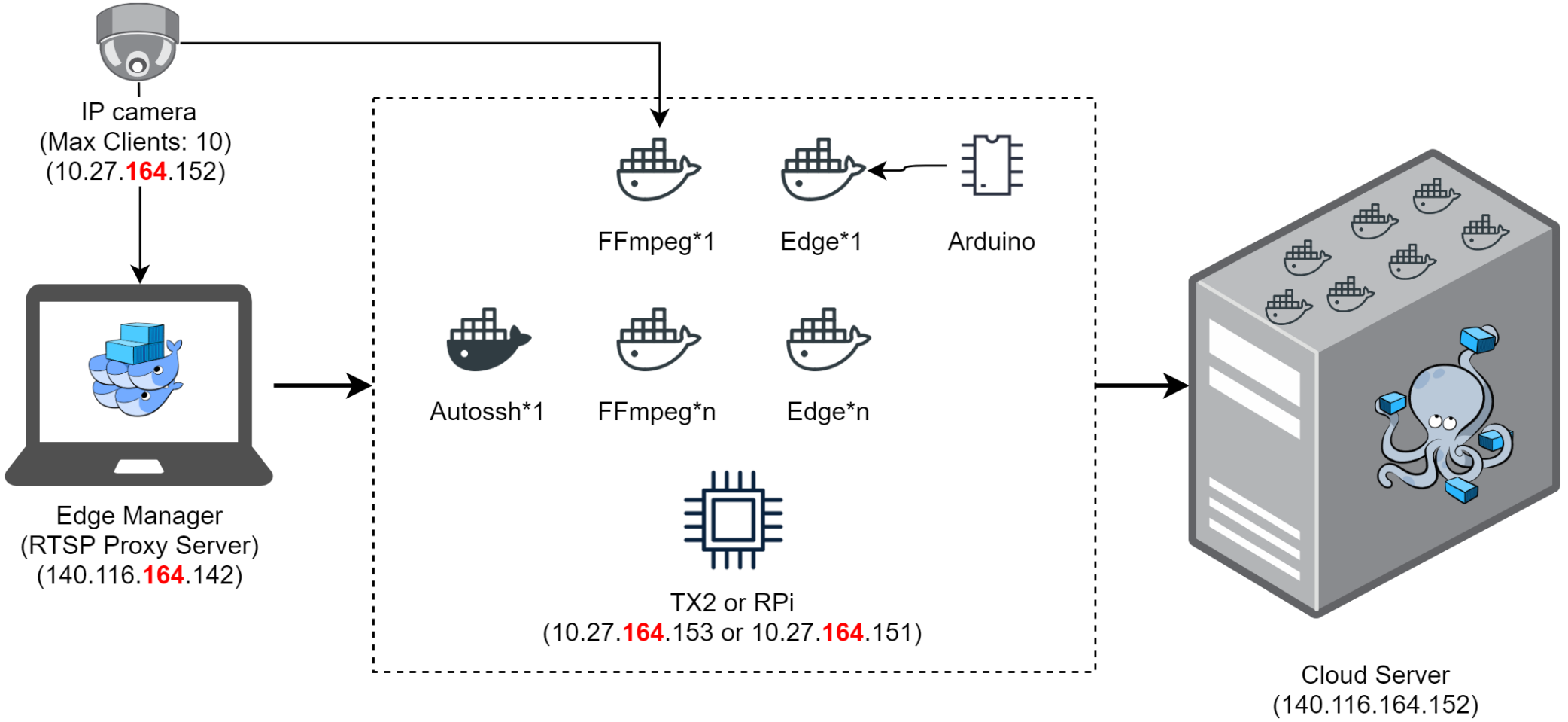


Performance of Cloud Server

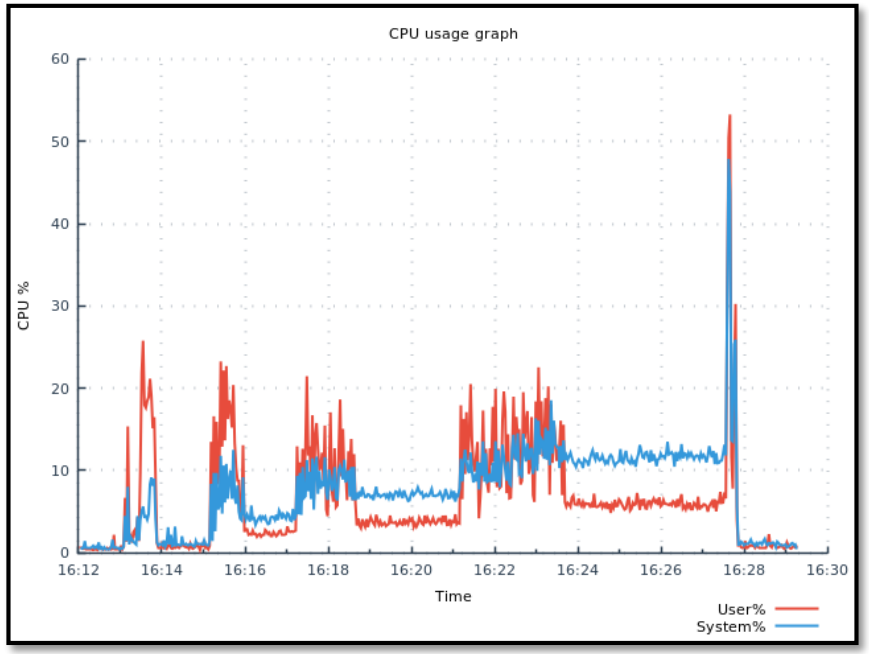
Table 4.4: CPU (%) and RAM (MB) usage of Cloud Server (average with 30 records)

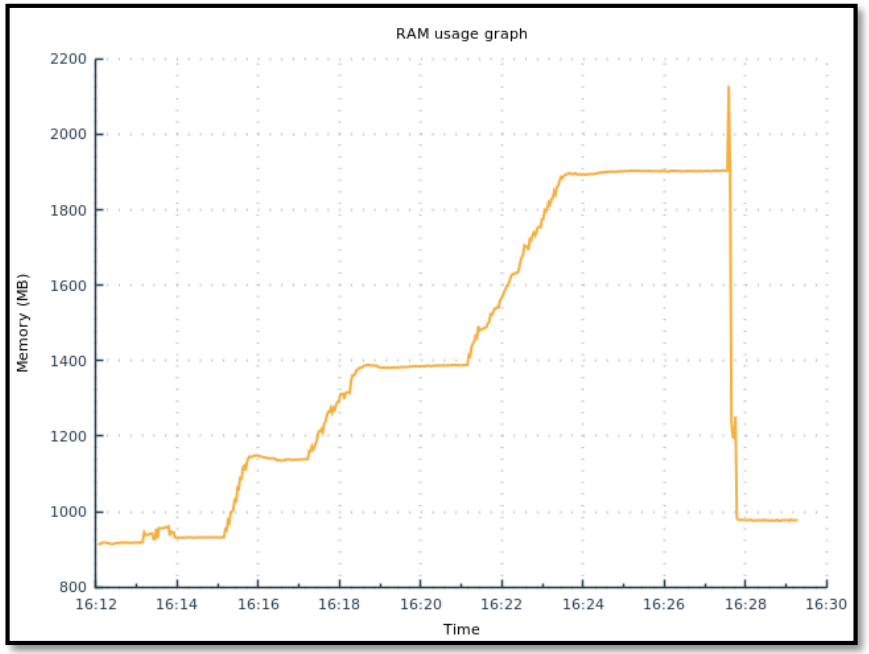
	nginx	nginx-rtmp	certbot	web	mongodb	redis	sshd
0 edge, 0 client	0.00% 28.67MB	0.00% 8.04MB	0.00% 3.67MB	0.14% 261.60MB	0.32% 41.88MB	0.09% 3.46MB	0.03% 23.27MB
1 edge, 0 client	0.00% 28.88MB	0.47% 8.15MB	0.00% 3.67MB	0.953% 280.58MB	0.56% 41.90MB	0.12% 3.47MB	0.30% 24.31MB
1 edge, 1 client	0.17% 28.88MB	0.46% 8.17MB	0.00% 3.67MB	0.27% 262.62MB	0.37% 41.96MB	0.13% 3.48MB	0.31% 24.29MB
1 edge, 10 clients	1.36% 30.71MB	2.32% 9.13MB	0.00% 3.67MB	0.24% 279.93MB	0.37% 43.00MB	0.11% 3.53MB	0.32% 24.35MB
40 edges, 0 client	0.00% 29.71MB	12.04% 29.34MB	0.00% 3.67MB	1.40% 288.66MB	0.32% 44.65MB	0.38% 3.77MB	9.57% 28.05MB
40 edges, 10 clients	1.48% 30.64MB	11.78% 43.42MB	0.00% 3.67MB	2.22% 289.64MB	0.37% 44.97MB	0.43% 3.89MB	9.37% 28.33MB

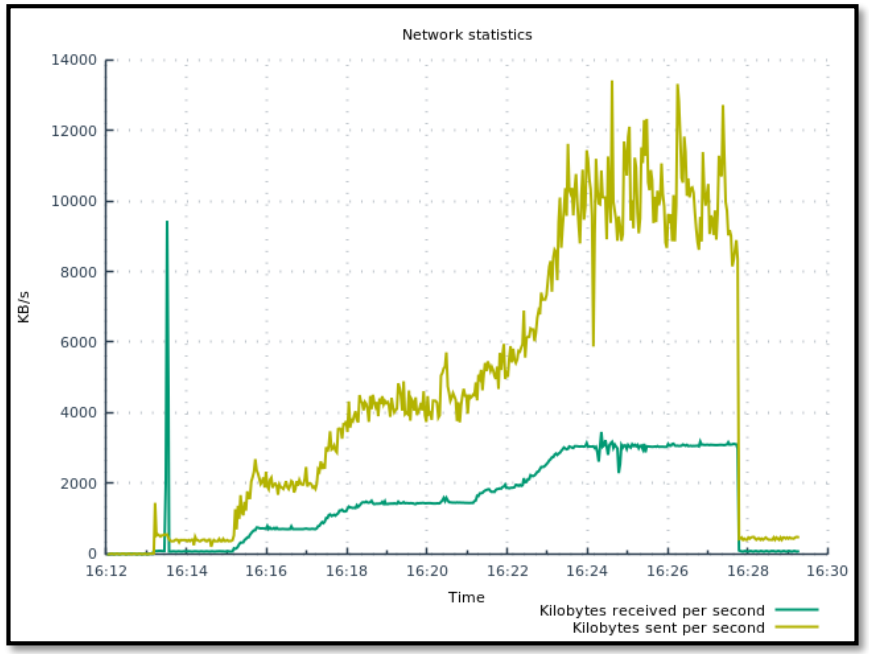
Performance of Edge Nodes



Performance of Edge Node (Nvidia TX2)

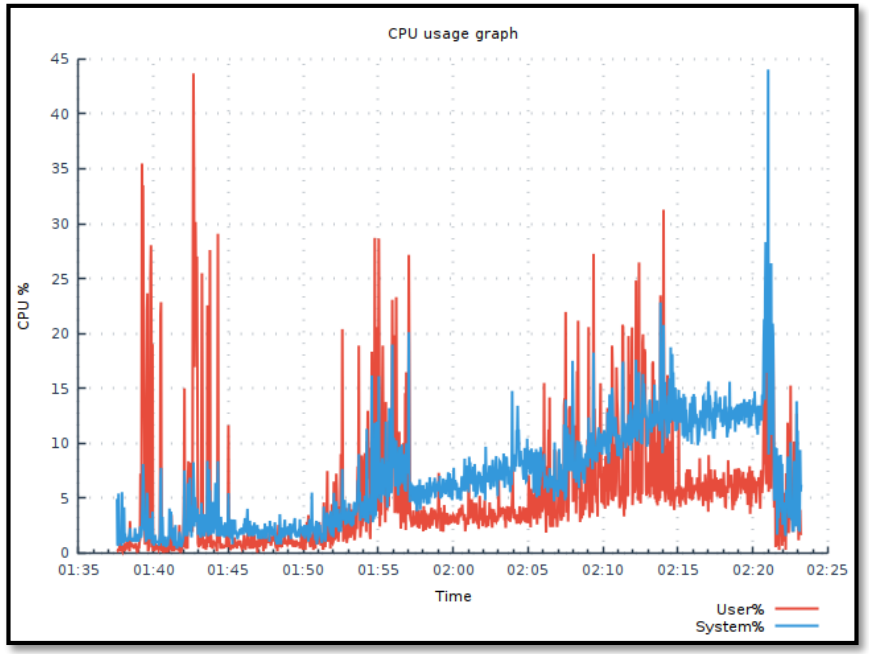


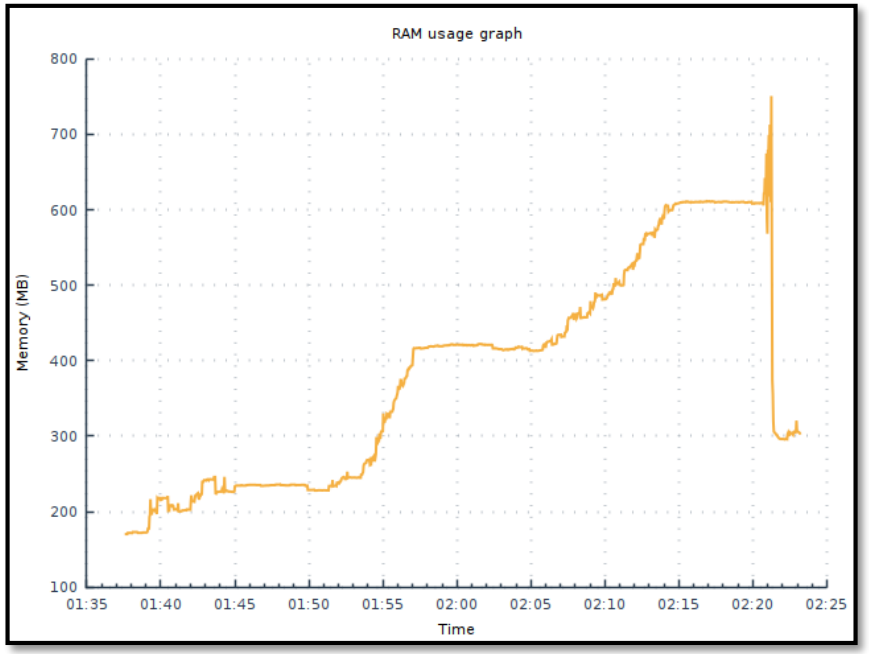


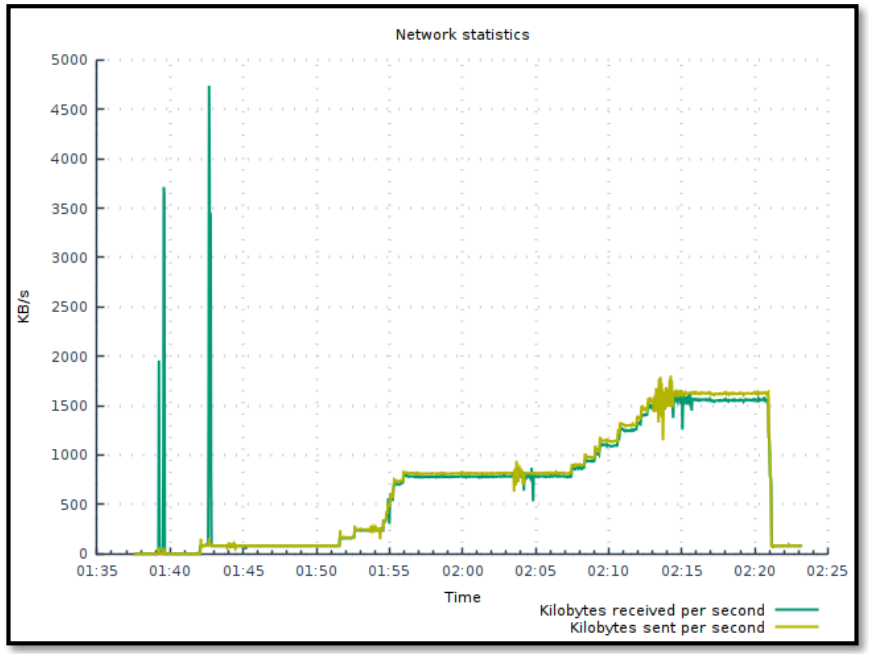


	Autossh	FFmpeg	Edge
16:13	1	1	1
16:15	1	10	10
16:17	1	20	20
16:21	1	40	40
16:28	1	1	1

Performance of Edge Node (RPi 3B)







	Autossh	FFmpeg	Edge
01:42	1	1	1
01:51	1	10	10
02:07	1	20	20
02:21	1	1	1

Outline

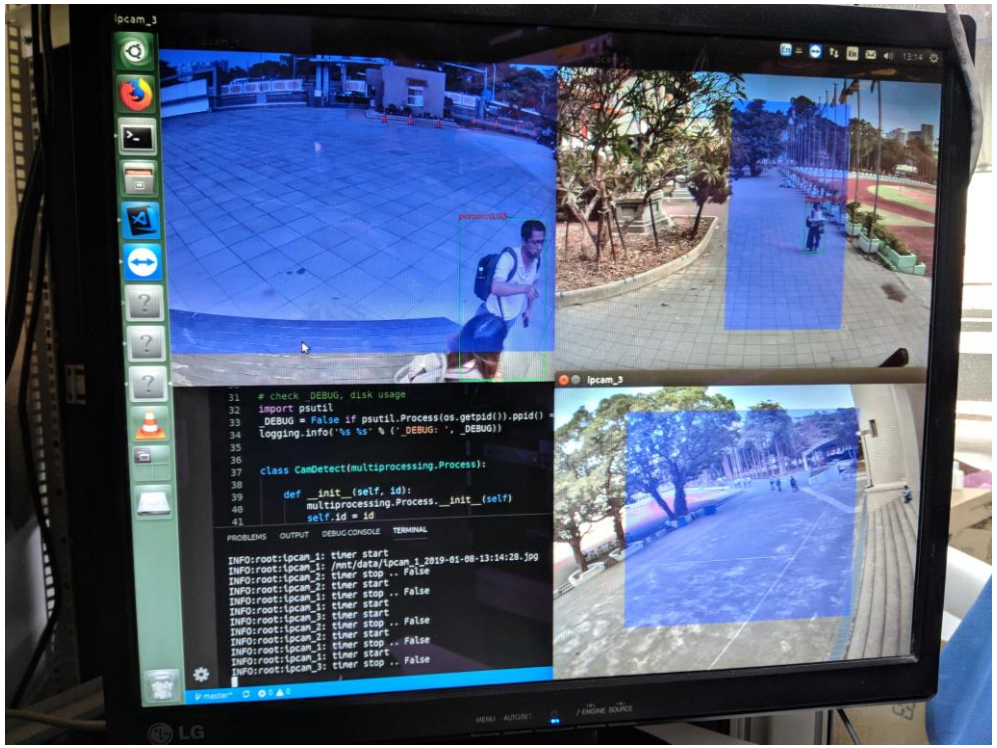
- Introduction
- Background and Related works
- System Design
- Experiment and Verification
- Conclusion and Future works

Conclusion

- Web-based streetlight management system
  - Lighting control (real-time)
  - Information of lamppost (real-time)
  - Historical information query API
- Secure Transmission (Cloud-Edge, Cloud-Browser)
- Containerize services in both Cloud and Edge

Future works

- Sensors management
- Intelligent applications
  - Smart lighting control mechanism
  - Real-time object detection (at Edge)
  - …

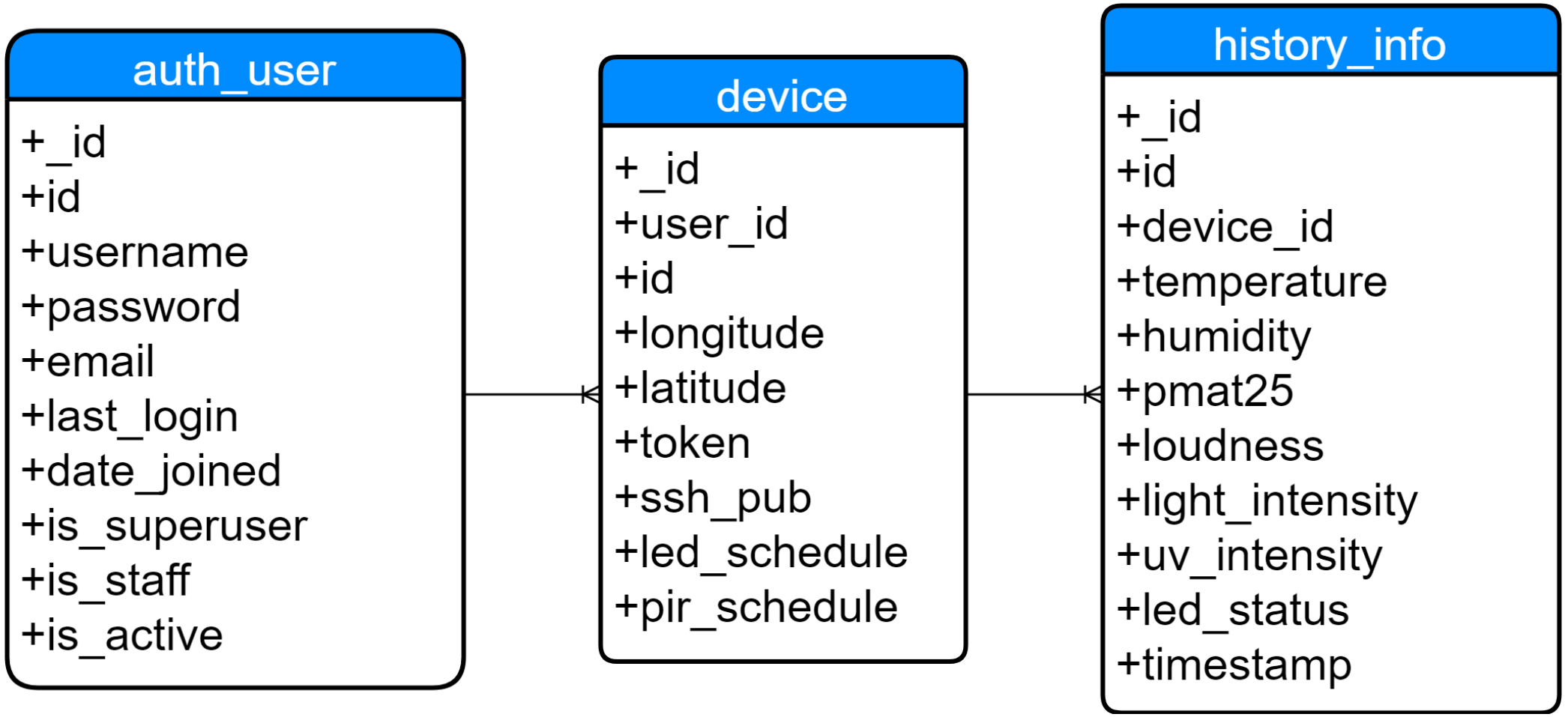


Thank you for listening !

Apendix

- Database (MongoDB)
- In-Memory Database (Redis)
- Edge.py
- Arduino.ino

Database (MongoDB)



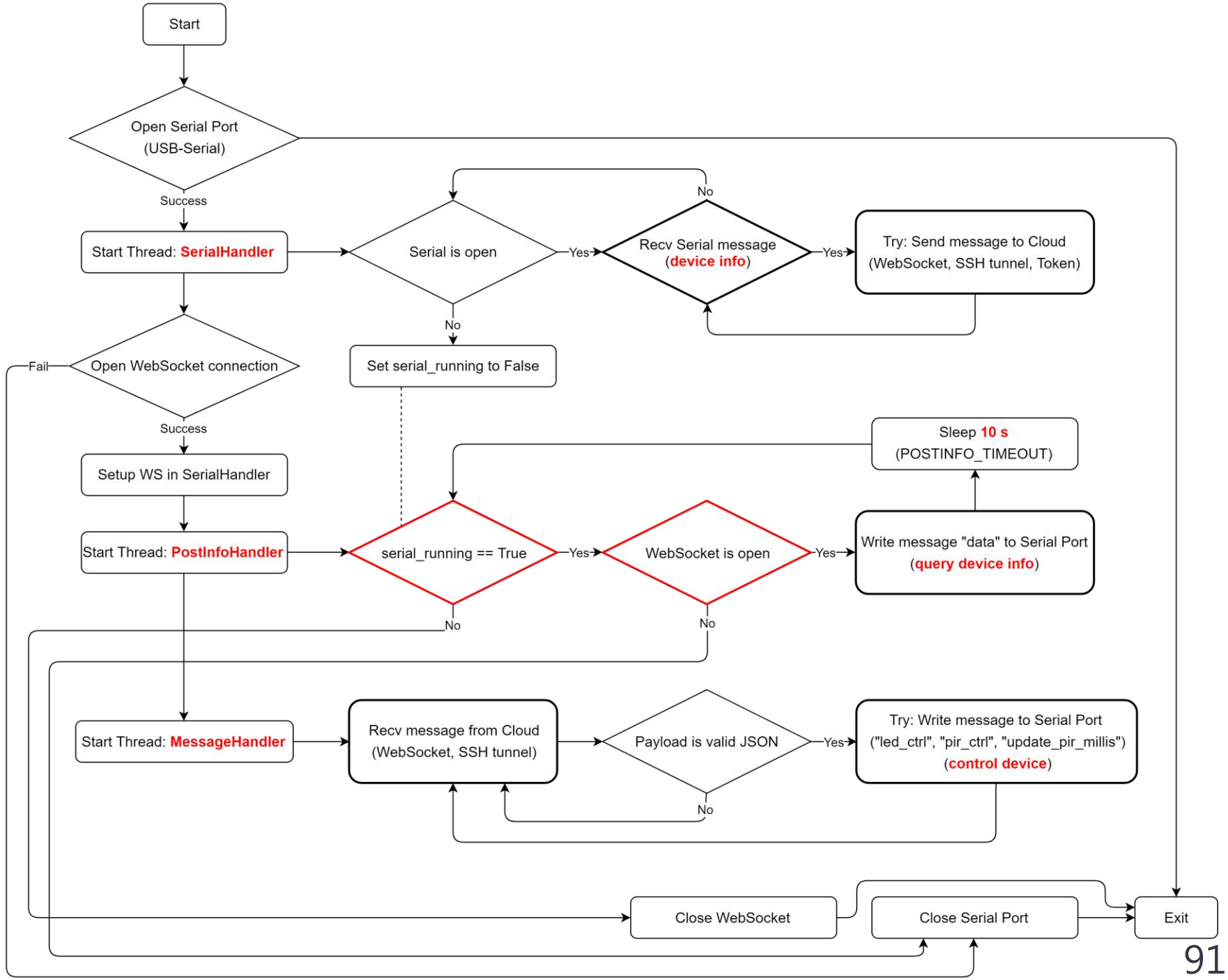
In-Memory Database (Redis)

Key	Value	TTL	Type
(session_key)	Session information	1 Day	Python pickle
(username)	Token for Open Data API	600 s	Python string
(Device ID)	Token for Edge services	-1 (∞)	Python string
(Device ID)_channel	WebSocket channel name of the Edge node	-1 (∞)	Python string
(Device ID)_info	Current information of the Edge node	30 s	Python pickle

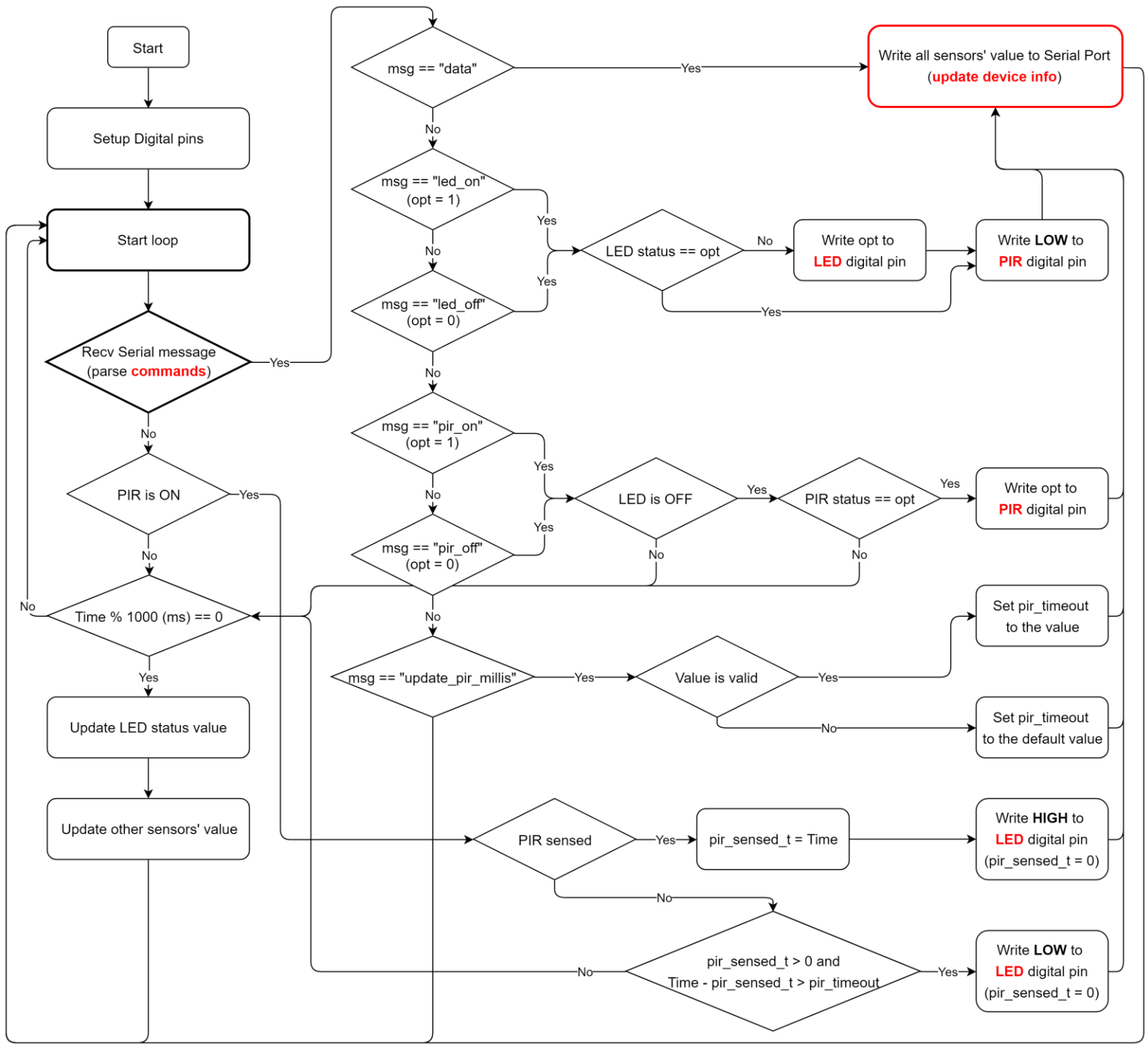
Edge.py

(Threads)

- **SerialHandler**
  (Arduino to Edge)
- **PostInfoHandler**
  (Edge to Arduino)
- **MessageHandler**
  (Cloud to Edge)



Arduino.ino


Start
Setup Digital pins
Start loop
Recv Serial message (parse commands)
Yes
No
PIR is ON
Yes
No
Time % 1000 (ms) == 0
No
Yes
Update LED status value
Update other sensors' value
msg == "data"
Yes
Write all sensors' value to Serial Port (update device info)
No
msg == "led_on" (opt = 1)
Yes
No
msg == "led_off" (opt = 0)
Yes
LED status == opt
No
Write opt to LED digital pin
Write LOW to PIR digital pin
Yes
No
msg == "pir_on" (opt = 1)
Yes
No
msg == "pir_off" (opt = 0)
Yes
LED is OFF
Yes
PIR status == opt
Yes
Write opt to PIR digital pin
No
No
No
msg == "update_pir_millis"
Yes
Value is valid
Yes
Set pir_timeout to the value
No
Set pir_timeout to the default value
PIR sensed
Yes
pir_sensed_t = Time
Write HIGH to LED digital pin (pir_sensed_t = 0)
No
No
pir_sensed_t > 0 and Time - pir_sensed_t > pir_timeout
Yes
Write LOW to LED digital pin (pir_sensed_t = 0)